CCCVs-Transfert

Rapport de projet

Caisse de compensation du canton du Valais
Ausgleichskasse Wallis

Service informatique

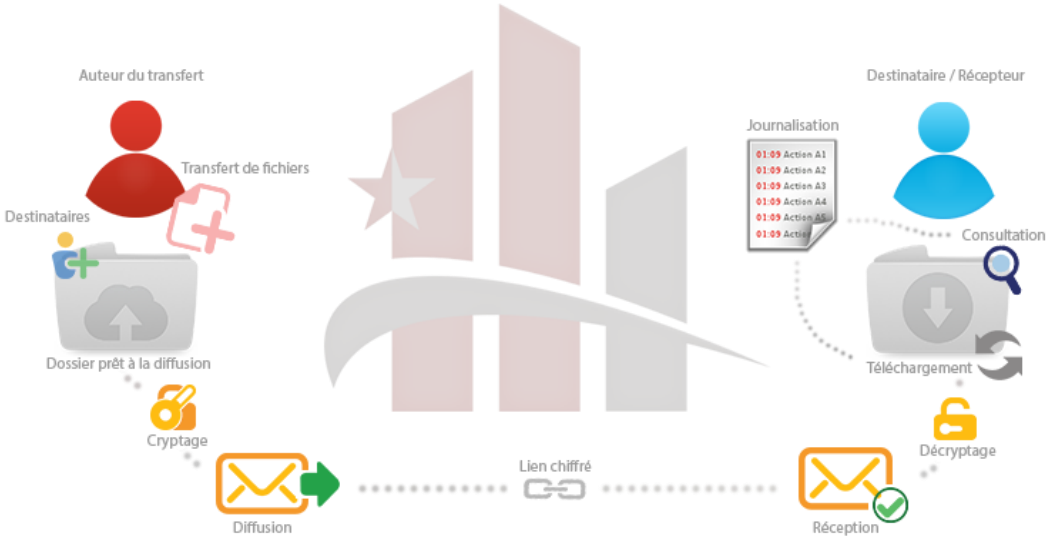
Dominique Roduit

27/03/2013

dominique.roduit@avs.vs.ch

CCCVs Transfert

Dominique Roduit – Ecole des métiers du Valais



Description

Le travail consiste en la réalisation d'une application internet riche basée sur la technologie Vaadin. Ce projet est constitué principalement d'une interface utilisateur sécurisée, bilingue et permettant le transfert de fichiers d'un utilisateur vers un ou plusieurs destinataires.

Les transferts sont représentés par des dossiers possédant chacun une durée de vie définie. Chaque utilisateur peut créer ses propres dossiers et choisir les contacts auxquels il souhaite partager les données. Les contacts agréés recevront un lien chiffré leur permettant l'accès aux fichiers du dossier qu'ils pourront télécharger l'un après l'autre ou sous forme d'archive. Toutes les actions entreprises sur les dossiers sont journalisées et résumées à l'auteur du transfert lorsque le dossier arrive au terme de sa durée d'accessibilité. Après quoi, les fichiers seront détruits.

Objectif

Développer une application internet riche, intuitive et moderne, implémentant un module de transfert de fichiers multiples pour permettre aux utilisateurs externes travaillant pour la Caisse de compensation de transférer des fichiers en interne et vice-versa. Une JavaDoc ainsi que plusieurs documentations et rapports de tests nécessaires à la reproductibilité du projet sont livrés en annexe.

Technologies utilisées

Les langages de développement utilisés

- Java — Application
- SQL — Base de données

Les librairies utilisées

- Vaadin — Interface utilisateur RIA
- EasyUploads — Transfert de fichiers

Réalisation

- Planification du travail à réaliser
- Analyse du cahier des charges
- Conception de la base de données et des schémas nécessaires à la compréhension de l'application
- Création de l'interface et de la navigation
- Développement des modules intégrés
- Réalisation de tests et rédaction des rapports
- Déploiement de l'application en production

Conclusion

Le cahier des charges a pu être réalisé pleinement et plusieurs fonctionnalités supplémentaires contribuant à renforcer l'intuitivité et l'efficacité de l'application ont également été implémentées lors du temps restant (par ex. la traduction de l'interface, la visualisation des actions journalisées sur les dossiers,

TABLE DES MATIERES

1. CAHIER DES CHARGES

Travail individuel dans le cadre de la production

Candidat :

Nom : Roduit Prénom : Dominique
Adresse : Rue de l'île 8 NPA, Localité : 3979 Grône
Adresse E-Mail : dominique.roduit@avs.vs.ch N° Tél prof.: 027 324 91 81

Entreprise :

Raison sociale : Caisse de compensation du canton du Valais
Adresse : Av. Pratifori 22 NPA, Localité : 1951 Sion

Chef du projet dans l'entreprise :

Nom : Lorétan Prénom : Yves
No Tél prof. : 027 324 91 72 No Fax : 027 324 94 68
Adresse E-Mail : yves.loretan@avs.vs.ch

Travail indivudel dans le cadre de la production :

Nom du projet : CCCVs-Transfert Durée (de 80 à 120 heures) : 120

Dates de réalisation *(obligatoirement entre le 18 février 2013 et le 26 avril 2013, le candidat doit suivre les cours professionnels durant cette période)*. Veuillez inscrire toutes les dates de réalisation: 25.02, 28.02, 01.03, 04.03, 07.03, 08.03, 11.03, 14.03, 15.03, 18.03, 21.03, 22.03, 25.03, 26.03, 27.03

Date de remise du projet à l'expert responsable *(Doit être la date du dernier jour du projet, cachet postal faisant foi)* : 27.03.2013

Date de défense du projet : 15.04.2013 à 08h30

Signature du chef du projet dans l'entreprise:

Lieu : Sion Date : 13.12.2012 Signature : ----------

Signature du candidat:

Lieu : Sion Date : 13.12.2012 Signature : ----------

1.1. INFORMATIONS SUR LE PROJET

Candidat

Nom	Roduit	Prénom	Dominique
Adresse	Rue de l'île 8	NPA, Localité	3979 Grône
Adresse E-mail	dominique.roduit@avs.vs.ch	N° Tél. prof.	027 324 91 81

Entreprise

Raison sociale	Caisse de compensation du canton du Valais		
Adresse	Av. Pratifori 22	NPA, Localité	1951 Sion

Chef du projet dans l'entreprise

Nom	Lorétan	Prénom	Yves
N° Tél. prof.	027 324 91 72	N° Fax	027 324 94 68
Adresse E-mail	yves.loretan@avs.vs.ch		

Experts chargés du suivi du projet

	Expert responsable	Expert adjoint
Nom / Prénom	David Joël	Genolet Antoine
N° Tél.	079 458 91 48	079 364 88 25
Adresse E-mail	joel.david@bluewin.ch	antoine.genolet@emvs.ch

Travail individuel dans le cadre de la production

Nom du projet	CCCVs-Transfert	Durée (de 80 à 120 heures)	120

Dates de réalisation

✓ 25.02.2013	✓ 08.03.2013	✓ 21.03.2013
✓ 28.02.2013	✓ 11.03.2013	✓ 22.03.2013
✓ 01.03.2013	✓ 14.03.2013	✓ 25.03.2013
✓ 04.03.2013	✓ 15.03.2013	✓ 26.03.2013
✓ 07.03.2013	✓ 18.03.2013	✓ 27.03.2013

Date de remise du projet à l'expert responsable

Date de défense du projet

Lundi, 15 avril 2013 à 08h30 – Salle de conférence de la CCCVs

1.2. DESCRIPTION DU PROJET

La Caisse de compensation du canton du Valais (CCCVs) s'occupe de différentes assurances sociales dont l'assurance vieillesse et survivants, l'assurance invalidité, les allocations pour perte de gain et les allocations familiales.

Elle compte actuellement plus de 150 employés répartis sur deux sites et désire se munir d'une interface permettant le transfert de fichiers via une application internet riche et sécurisée, basée sur la technologie Vaadin.

Cette application doit être accessible à l'extérieur du réseau interne de la caisse de compensation pour permettre notamment, de manière simple et intuitive, le transfert de fichiers volumineux entre l'internet et l'intranet, le but étant de mettre ces données à disposition d'une liste de destinataires définie par l'auteur du transfert.

Pour garantir la sécurité, toutes les opérations et données relatives aux utilisateurs seront journalisées et les transferts sécurisés par un certificat SSL. Les fichiers plus vieux que 10 jours seront automatiquement supprimés du serveur.

La mise en place de l'infrastructure du réseau ne fait pas partie de mes tâches qui seront uniquement de développer la partie application. Par conséquent, toute l'infrastructure permettant l'exécution du projet sera prête avant le début du développement.

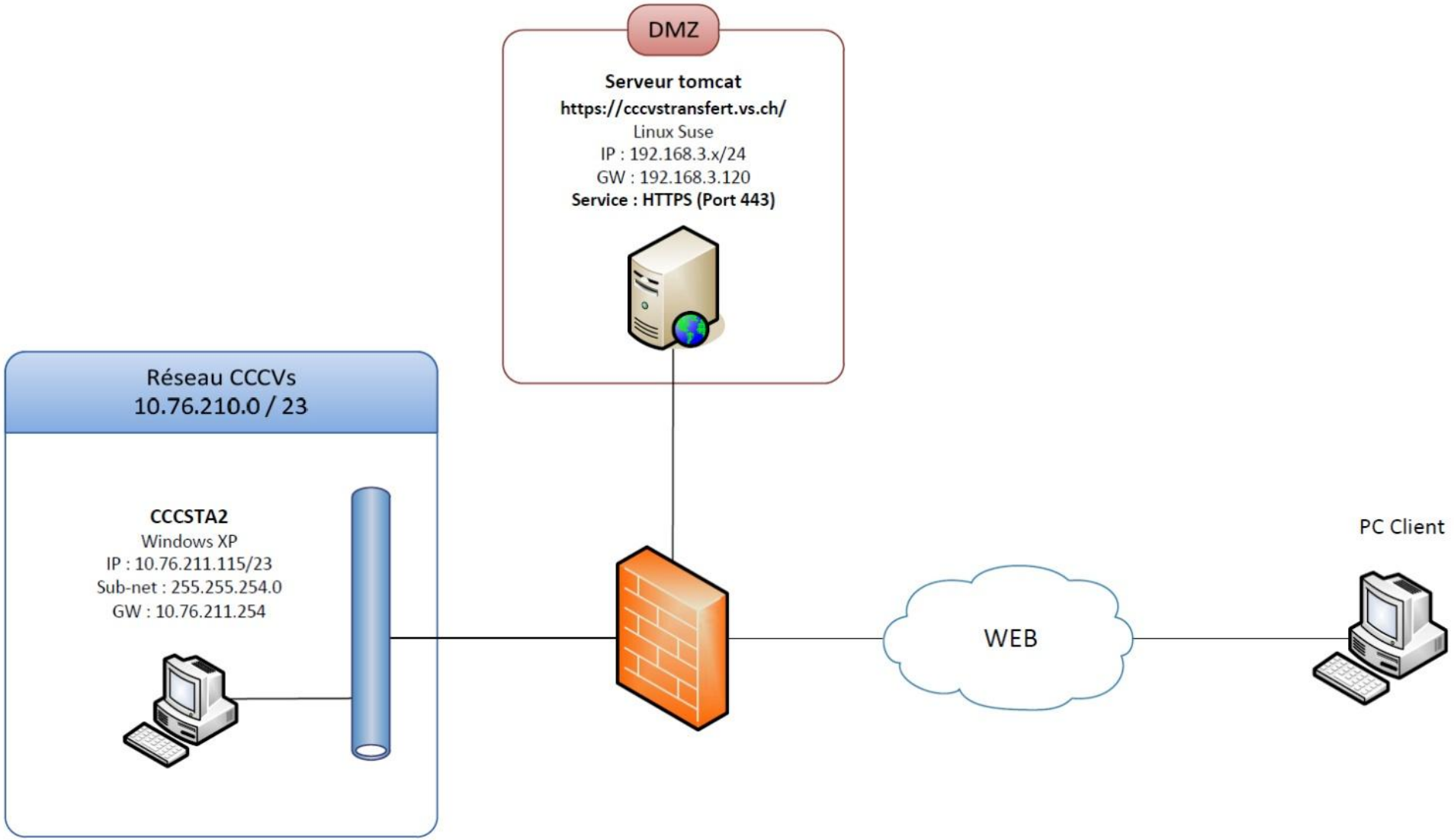
1.2.1. POURQUOI CE PROJET

Afin de permettre à la Caisse d'être autonome, de simplifier le transfert de fichiers et de pouvoir apporter les développements nécessaires à la Caisse. Cela, pour offrir un service personnalisé, qui sera utilisé par des personnes externes, mais travaillant pour la Caisse.

1.3. INFRASTRUCTURE

- ✓ Base de données : DB2 pour AIX *(07.02.2013 : Modification pour MySQL)*
- ✓ Outils de développement JAVA : Rational® Business Developer™
- ✓ Librairies VAADIN
- ✓ Serveur linux qui héberge l'application
- ✓ Serveur Tomcat

1.4. SCHÉMA RÉSEAU



1.5. DÉTAILS SUR LA RÉALISATION

1.5.1. PARTIE 1

1	**Conception du schéma de navigation et validation auprès du responsable** Concevoir un schéma de la navigation qui présente un ordre/une disposition des pages pour la navigation, dans le but de valider cette partie avant de s'investir dans le code.
2	**Conception du schéma de base de données de l'application** Schématisation de la base de données (journalisation) à l'aide de MySQL Workbench, il faudra définir les tables utiles pour le stockage des informations, trouver des noms cohérents pour chacune d'elle, puis créer les tables sur la base de ce schéma. La base de données sera de type DB2.
3	**Conception du design de l'application** Mise en place du graphisme de l'application à l'aide des templates Vaadin qui permettent un rendu similaire sur l'ensemble des navigateurs, anciens ou récents. Il m'est demandé d'innover, de rendre l'interface la plus moderne possible tout en respectant l'identité graphique de la caisse de compensation.
4	**Navigation en HTTPS : installation du certificat SSL** Jusqu'à la fin du TPI et tant que l'application restera en phase de développement, elle sera sécurisée par le protocole HTTPS, à l'aide d'un certificat gratuit valide pour le navigateur.
5	**Reconnaissance de la provenance des utilisateurs (Réseau interne / externe)** Aucune authentification n'est demandée aux utilisateurs de l'application, car elle doit être disponible par n'importe qui en interne comme en externe. Pour pouvoir identifier les utilisateurs, il est donc impératif de savoir d'où ils proviennent pour ensuite récupérer des informations qui peuvent servir à leur identification.
6	**Transfert multiple de fichiers vers le serveur** Ceci est la tâche principale de l'application qui doit permettre à l'utilisateur de pouvoir transférer un ou plusieurs fichiers vers le serveur. La taille maximum des fichiers en envoi simultané ne doit pas dépasser 1 Go, car un timeout limite le temps d'exécution d'une page.
7	**Développement de la page de téléchargement (récupération) des fichiers.** Cette page permettra à l'auteur du transfert ainsi qu'à toutes les personnes à qui sont destinés les fichiers de pouvoir les télécharger. Elle devra être accessible grâce à un paramètre crypté dans l'URL pour rendre les données confidentielles.
8	**Envoi d'un mail contenant un lien sécurisé pour télécharger les fichiers** Le lien sécurisé pour l'accès à la page de téléchargement sera ensuite envoyé à tous les destinataires. Selon le choix de l'auteur du transfert, soit un lien différent sera envoyé à chaque destinataire pour améliorer la traçabilité du système, soit un même lien sera envoyé à tous.
9	**Suppression automatique des fichiers.** L'auteur du transfert peut choisir combien de temps il met à disposition les fichiers qu'il transfère. Lorsque le nombre de jours mentionné (de 1 à 10) est écoulé, les fichiers sont automatiquement supprimés du serveur grâce à un script automatique exécuté quotidiennement
10	**Journalisation du trafic et des opérations des utilisateurs** Toutes les informations concernant les transferts, les téléchargements et les utilisateurs seront enregistrées dans une base de journalisation pour nous permettre de détecter un éventuel problème et d'avoir une traçabilité de toutes les opérations exécutées.
11	**Tests de l'application** L'application sera testée depuis plusieurs postes et plusieurs réseaux. Un protocole de test sera établi.

Les tâches de la partie deux du cahier des charges sont facultatives et ne seront effectuées que si toutes celles de la partie une se révèlent entièrement fonctionnelles.

1.5.2. PARTIE 2

1	**Envoi d'un mail d'information avant toutes suppressions de fichiers.** Un jour avant la suppression automatique des fichiers, un script exécuté automatiquement sur le serveur envoie un mail à l'auteur du transfert ainsi qu'aux personnes à qui sont destinés ces fichiers pour les avertir de leur suppression prochaine.
2	**Création d'une interface « admin » offrant la possibilité d'analyser toutes les informations journalisées.** Pour permettre une analyse et une accessibilité plus agréable aux informations journalisées, il serait intéressant de développer une zone sécurisée uniquement accessible par mot de passe.
3	**Affichage de statistiques sur les opérations journalisées dans la zone « admin ».** Dans la zone accessible uniquement par mot de passe, il serait également intéressant d'afficher des statistiques sous forme graphique sur les transferts effectués sur une période donnée.

1.5.3. PERSONNES IMPLIQUÉES DANS LA RÉUSSITE DU PROJET

Mr. Philippe Arcudi	CCCVs	Chef du service informatique Responsable système exploitation
Mr. Bernard Dubuis	CCCVs	Adjoint de Mr. Arcudi Responsable développement
Mr. Yves Lorétan	CCCVs	Responsable sécurité et équipements Chef du projet dans l'entreprise

1.6. DOCUMENTS À REMETTRE

Le dernier jour de la période allouée au projet, le candidat remettra en trois exemplaires :

- ✓ Toute la documentation papier nécessaire (pas de disquettes) à la compréhension et à la reproductibilité du travail final.
- ✓ La planification du projet.
- ✓ Le journal de bord du projet.

Un dossier doit parvenir le jour même (le cachet de la poste faisant foi) aux deux experts évaluant le projet.

Un autre dossier doit parvenir au chef responsable du projet dans l'entreprise. Celui-ci doit l'analyser et le remettre ensuite (avec les corrections) à l'expert responsable.

1.7. LOGICIELS UTILISÉS

Les logiciels suivants ont été utiles à la réalisation du projet :

OpenOffice, Microsoft Office 2010/2013	Rédaction des rapports
Mozilla Firefox + Firebug	Recherche d'informations / Inspection DOM
Internet Explorer / Google Chrome / Mozilla Firefox	Tests de compatibilité multi-navigateurs
MS Project	Réalisation de la planification
Rational Business Developer + Vaadin	Développement Java
phpMyAdmin	Gestion de la base de données
Microsoft Visio	Conception des schémas
MySQL Workbench	Schématisation de la base de données
GIMP	Retouches et conception d'images

Séance : CCCVs-Transfert – Démarrage du projet — 1.8. Procès-verbal

Présents :	Joël David, Antoine Genolet, Philippe Arcudi	**Excusés :**	Yves Lorétan
Copie :	yves.loretan@avs.vs.ch	**Protocole :**	
Lieu :	CCCVs - Sion		
Date :	28.02.13	**Heure :**	10h00 à 11h00

Ordre du jour :

1. Modalités – Spécifications administratives
2. Documentation à rendre
3. Planification des séances
4. Divers

1. Modalités – Spécifications administratives	QUI	Délai
Distinction des meilleurs TPI • Rédiger un bref rapport du TPI sur une page, qui montre l'essentiel en un coup d'œil.	JOD	
Tout mettre dans la documentation, en n'oubliant jamais d'insérer les raisonnements que l'on a fait pour arriver à une conclusion, un choix. Toujours bien expliquer nos choix, mentionner les avis du client et la validation par celui-ci.	AGE	
Prioriser le cahier des charges avant tout le reste car les tâches annexes qui n'en font pas parties ne seront pas évaluées !		
Défense ➔ maximum 1h *(temps annoncé au début de la présentation)* • Présentation power point ➔ analyse, choix, ... • Démonstration de l'application en l'état au 27.03.2013	AGE	

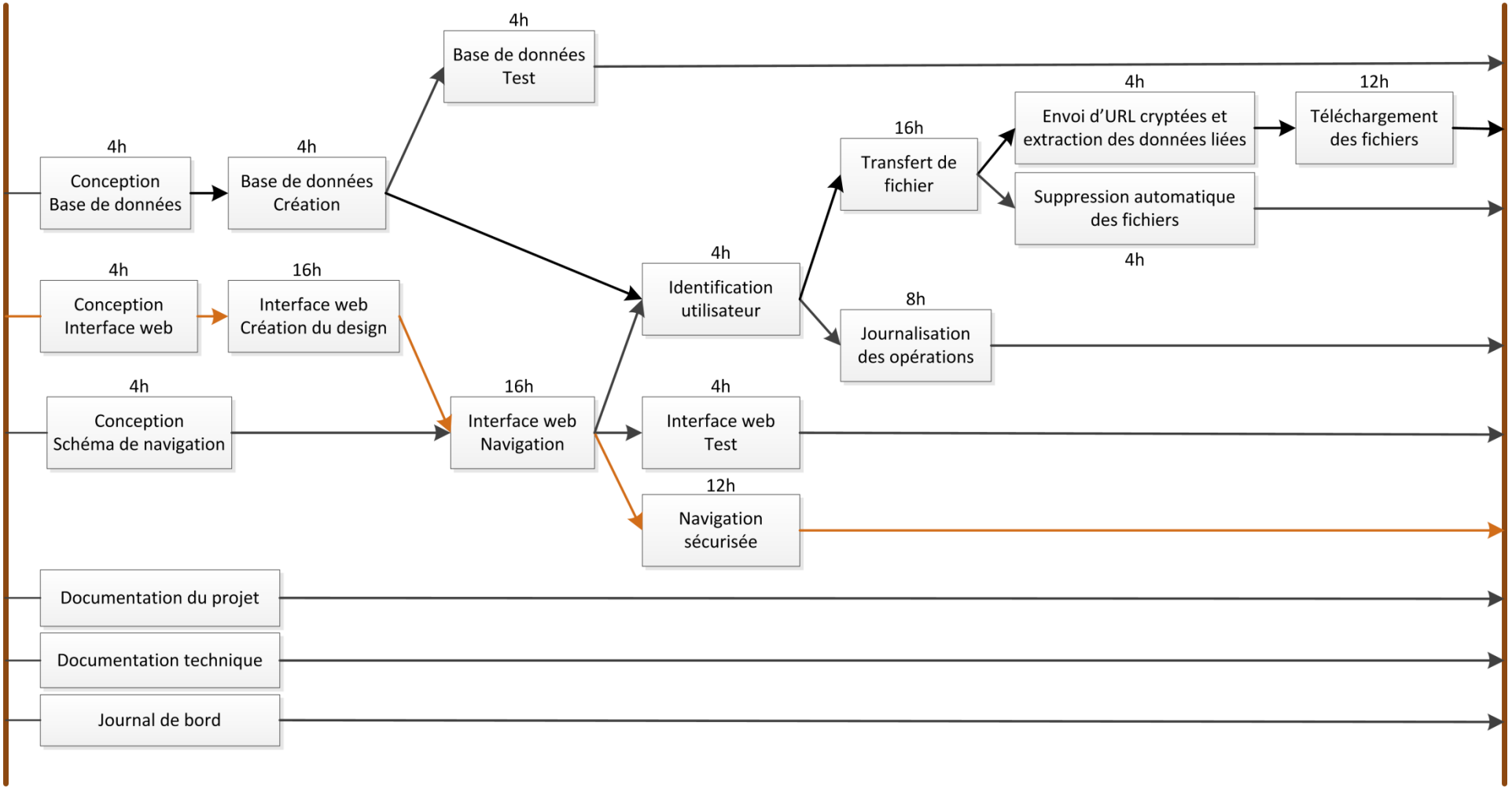
2. Documentation à rendre	QUI	Délai

Minimum des documents à rendre
• Journal de bord (Indiquant également toutes les séances, problèmes, choix, décisions, aides externes)
• Documentation de projet (administrative)
  ○ Planification initiale / réelle
• Documentation technique
• Annexes
  ○ Documentation utilisateur
  ○ Rapport de tests
  ○ Schémas des classes (UML)
  ○ Tout le code personnel		

3. Planification des séances	QUI	Délai
Visite intermédiaire : Lundi 11 mars 2013 à 10h30 Défense du projet : 15 avril 2013 à 08h30 ou 17h00 (A valider avec YLO et confirmer aux experts)		

4. Divers	QUI	Délai
Envoyer la planification initiale aux experts Résultats pas avant le début Juillet Faire des Backups régulier du projet Versioning du projet	DRO	

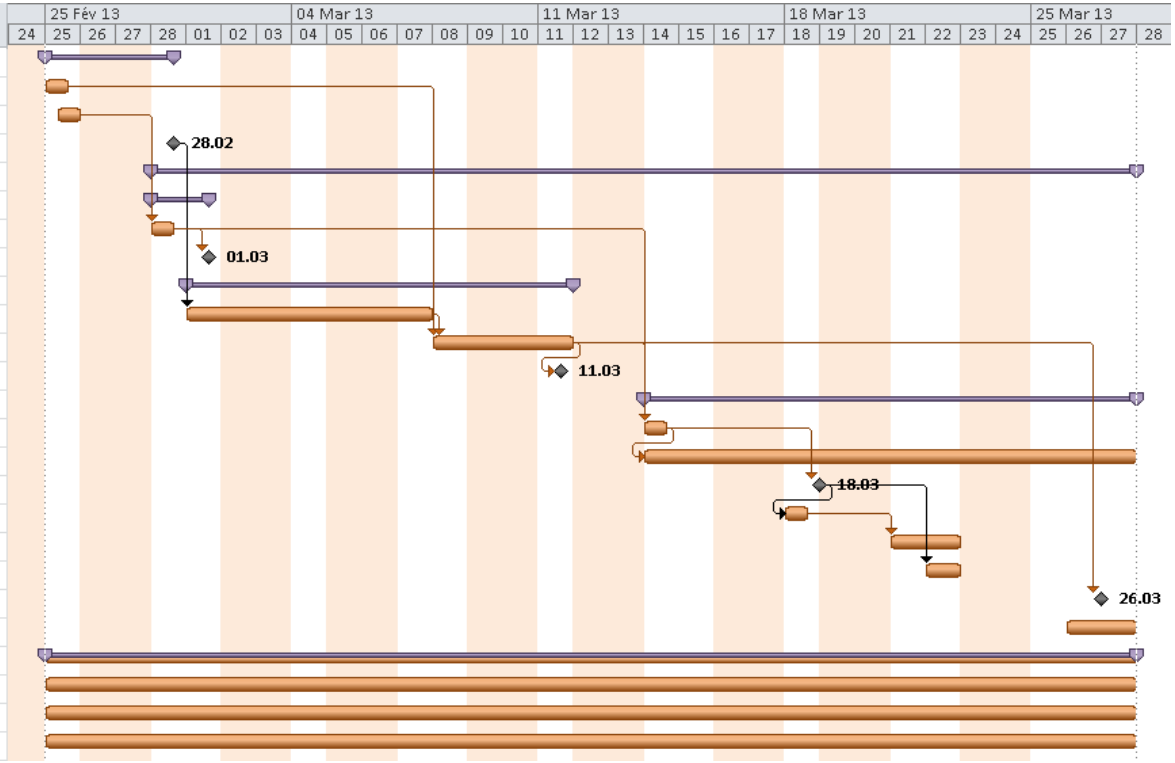
2. SCHÉMA PERT



3. PLANIFICATION DU PROJET

Durées des tâches estimées avant le début du projet (22.02.2013)

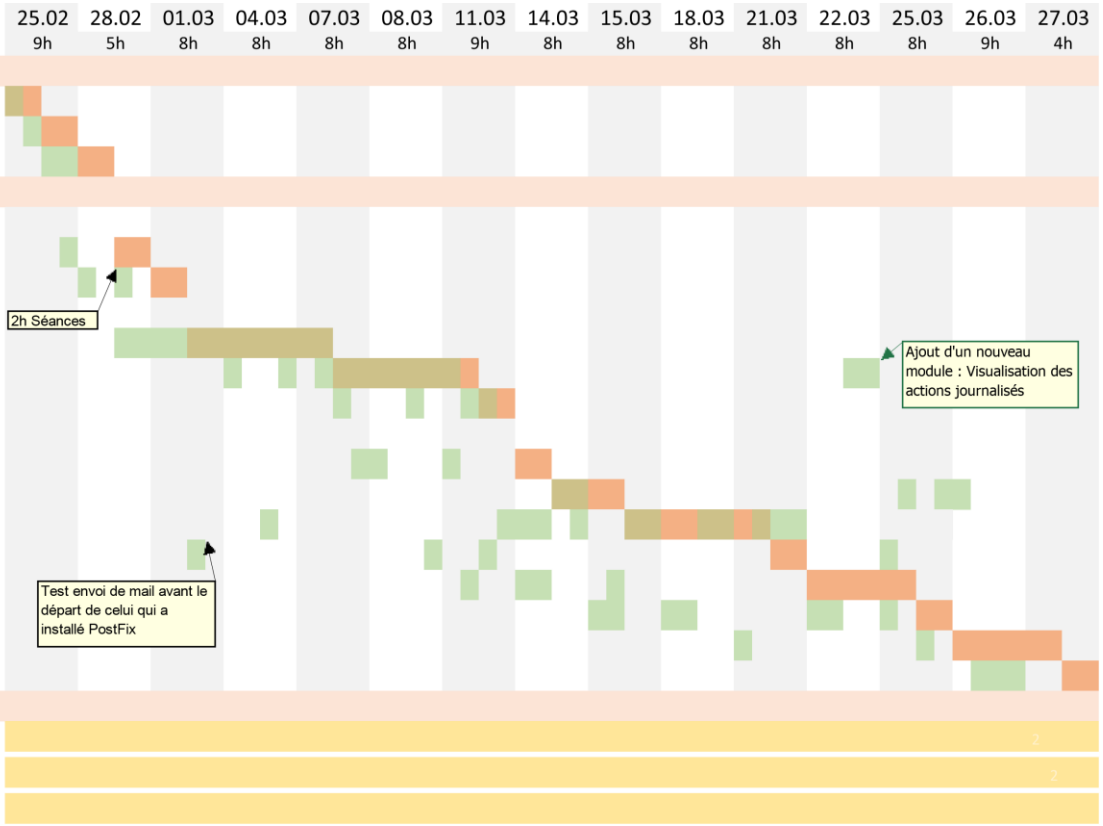
Nom de la tâche	Travail	Début	Fin
1 Conception	**12 h**	**Lun 25.02.13**	**Jeu 28.02.13**
1.1 Schéma de navigation	4 h	Lun 25.02.13	Lun 25.02.13
1.2 Base de données	4 h	Lun 25.02.13	Lun 25.02.13
1.3 Interface web	4 h	Jeu 28.02.13	Jeu 28.02.13
2 Réalisation	**108 h**	**Jeu 28.02.13**	**Mer 27.03.13**
2.1 Base de données	**8 h**	**Jeu 28.02.13**	**Ven 01.03.13**
2.1.1 Création	4 h	Jeu 28.02.13	Jeu 28.02.13
2.1.2 Test	4 h	Ven 01.03.13	Ven 01.03.13
2.2 Interface web	**36 h**	**Ven 01.03.13**	**Lun 11.03.13**
2.2.1 Création du design	16 h	Ven 01.03.13	Jeu 07.03.13
2.2.2 Navigation	16 h	Jeu 07.03.13	Lun 11.03.13
2.2.3 Test	4 h	Lun 11.03.13	Lun 11.03.13
2.3 Modules spécifiques	**64 h**	**Jeu 14.03.13**	**Mer 27.03.13**
2.3.1 Identification utilisateur	4 h	Jeu 14.03.13	Jeu 14.03.13
2.3.2 Journalisation des opérations	8 h	Jeu 14.03.13	Mer 27.03.13
2.3.3 Transfert de fichier	16 h	Jeu 14.03.13	Lun 18.03.13
2.3.4 Envoi d'URL cryptées et extraction des données liées	4 h	Lun 18.03.13	Lun 18.03.13
2.3.5 Téléchargement des fichiers	12 h	Jeu 21.03.13	Ven 22.03.13
2.3.6 Suppression automatique des fichiers	4 h	Ven 22.03.13	Ven 22.03.13
2.3.7 Navigation sécurisée	12 h	Lun 25.03.13	Mar 26.03.13
2.3.8 Rapport de test	4 h	Mar 26.03.13	Mer 27.03.13
3 Documentation	**120 h**	**Lun 25.02.13**	**Mer 27.03.13**
3.1 Documentation du projet	40 h	Lun 25.02.13	Mer 27.03.13
3.2 Documentation technique	40 h	Lun 25.02.13	Mer 27.03.13
3.3 Journal de bord	40 h	Lun 25.02.13	Mer 27.03.13



4. UTILISATION RÉELLE DU TEMPS IMPARTI

Différences avec la planification initiale.

Tâches	Initial	Réel	Diff.
	120h	116h	4h
1 Conception	**12h**	**7h**	**5h**
1.1 Schéma de navigation	4h	2h	**2h**
1.2 Base de données	4h	2h	**2h**
1.3 Interface web	4h	3h	**1h**
2 Réalisation	**108h**	**105h**	**3h**
2.1 Base de données	**8h**	**4h**	**4h**
2.1.1 Création	4h	2h	**2h**
2.1.2 Test	4h	2h	**2h**
2.2 Interface web	**36h**	**38h**	**-2h**
2.2.1 Création du design	16h	17h	**-1h**
2.2.2 Navigation	16h	17h	**-1h**
2.2.3 Test	4h	4h	**0h**
2.3 Modules spécifiques	**64h**	**63h**	**1h**
2.3.1 Identification utilisateur	4h	3h	**1h**
2.3.2 Journalisation des opérations	8h	9h	**-1h**
2.3.3 Transfert de fichier	16h	20h	**-4h**
2.3.4 Envoi d'URL cryptées et extraction des données liées	4h	4h	**0h**
2.3.5 Téléchargement des fichiers	12h	4h	**8h**
2.3.6 Suppression automatique des fichiers	4h	12h	**-8h**
2.3.7 Navigation sécurisée	12h	4h	**8h**
2.3.8 Rapport de test	4h	7h	**-3h**
3 Documentation	**120h**		
3.1 Documentation du projet	40h		
3.2 Documentation technique	40h		
3.3 Journal de bord	40h		



Etat au 27.03.2013. = Planification initiale = Planification réelle

J'en conclu que je n'ai pas su raisonnablement évaluer le temps nécessaire à l'accomplissement des tâches du projet. J'ai surestimé pratiquement tous les objectifs, à l'exception de la « suppression automatique des fichiers » qui a été réalisée en 1 jour de plus que ce qui était initialement prévu.

Cependant, toutes les tâches ont été réalisées dans le temps accordé. La partie optionnelle du cahier des charges (non représentée) a également été exécutée à quelques détails près *(cf. Documentation technique, chapitre 16)*.

5. STRUCTURE MISE EN PLACE DANS L'ENTREPRISE

Pour permettre la réalisation du projet, plusieurs tâches devaient être accomplies pour préparer les différents outils, avant la phase de développement.

Voici une liste succincte des installations et principales configurations effectuées :

Serveur Linux Suse Enterprise Edition 11 *Hostname : prdWTransfert* *IP : 10.76.210.172* *User : root* *Pass : K********	Apache Tomcat 7	Version : 7.0.35 Port d'exécution : 8080 Répertoire : /opt/tomcat7/ Manager : admin, admin
	MySQL	Version : 5.6.10 Port d'exécution : 3306 User : admin, K*******
	Serveur SMTP	Port d'exécution : 25

6. LIBRAIRIE VAADIN

6.1. QU'EST-CE QUE VAADIN ?

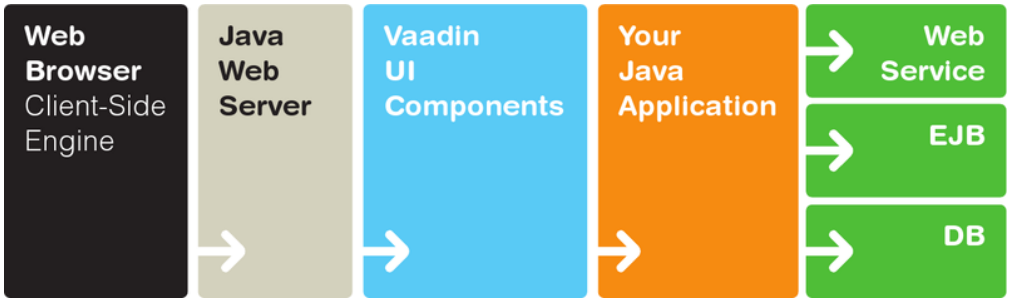
Vaadin est un framework Java open source qui permet la création d'applications Internet riches (RIA, Rich Internet application). Contrairement aux bibliothèques JavaScript, cette librairie dispose d'une architecture côté serveur, ce qui signifie que la majorité d'une application Vaadin est écrite en Java et que très peu de code CSS, HTML et Javascript sont ajoutés. Cela se fait uniquement lorsque nous devons manipuler nos pages web de manière plus avancées. Google Web Toolkit (GWT) est utilisé côté client, afin d'assurer une expérience utilisateur riche et interactive.

Vaadin possède également une grande collection de composants utiles à la construction d'interfaces utilisateurs. Celles-ci sont composées dans la plupart des cas de boutons, tableaux, arbres et layout. Chaque composant utilise des événements (events), écouteurs (listeners) et liaisons de données pour communiquer avec les autres.

L'architecture basée sur des composants en Java aide à construire des applications facilement modulables et remaniées en fonction des besoins. Un outil de conception visuel intégré dans un plugin eclipse permet également la construction rapide d'interfaces web.

6.2. VUE D'ENSEMBLE

Le cœur de Vaadin est la bibliothèque Java, conçue pour faciliter la création et la maintenance d'interfaces web de haute qualité. L'idée principale du modèle de programmation Vaadin consiste à pouvoir programmer un peu comme est programmée n'importe quelle application Java traditionnelle avec des outils classiques tels qu'AWT, Swing ou SWT.



Source : http://bit.ly/UXLKc0

Le schéma ci-dessus illustre l'architecture de base d'une application web dotée de Vaadin, application qui consiste en une librairie coté serveur et un moteur coté client qui s'exécute dans le navigateur comme un programme JavaScript qui envoie les interactions de l'utilisateur au serveur qui lui-même retourne les résultats au client.

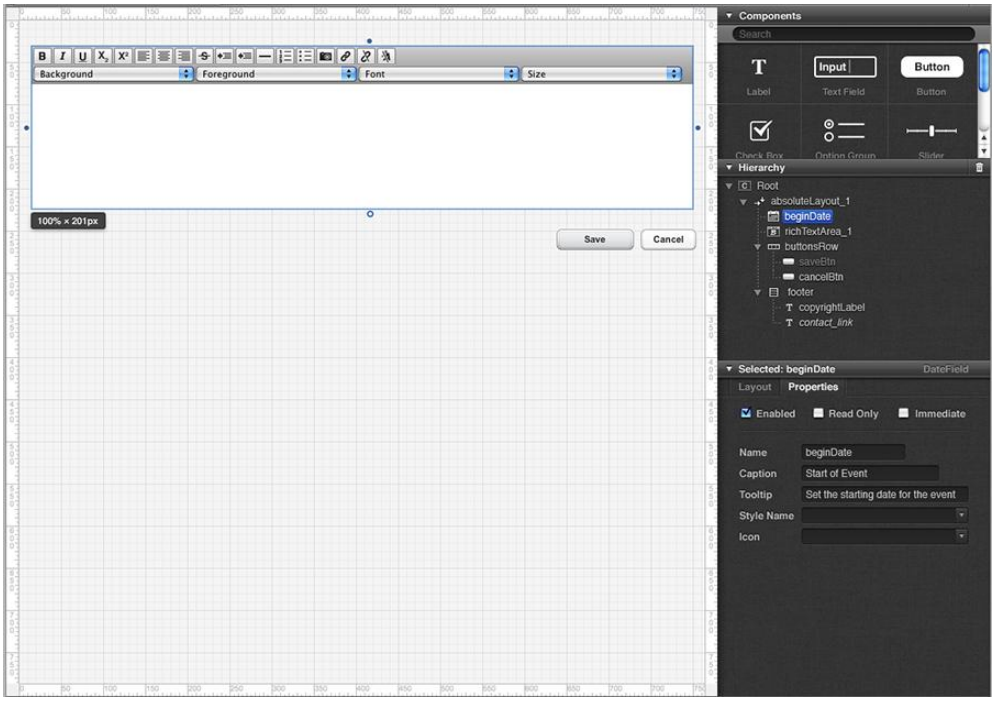
Comme l'HTML, le JavaScript et les autres technologies du navigateur sont invisibles à la logique de l'application, nous n'avons que très peu à nous soucier de la compatibilité multi-navigateur, notre code sera interprété et affiché plus ou moins de la même manière sur les différents navigateurs actuels.

Puisque le moteur côté client est exécuté comme du JavaScript dans le navigateur, aucun plugin n'est nécessaire pour utiliser les applications construites avec Vaadin.

6.3. Développement avec Vaadin

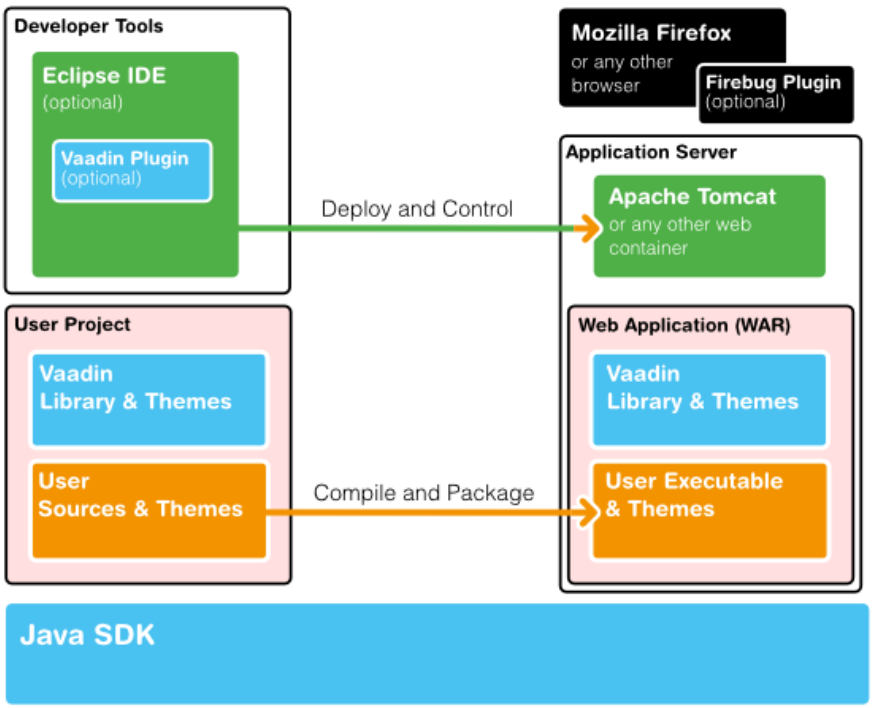
6.3.1. Environnement

Le développement de la partie graphique de l'interface utilisateur avec la librairie Vaadin peut s'écrire à la main ou, pour des applications relativement simples, à l'aide de l'outil « Visual Designer » présenté ci-dessous.



Capture d'écran du « Visual Designer »

6.3.2. OUTILS ET PROCESSUS DE DÉVELOPPEMENT



Source : http://bit.ly/Y4C4H9

Le schéma ci-dessus illustre simplement le processus standard de développement d'une application Vaadin à travers les outils de développements Java usuels. Les applications Vaadin sont développées comme n'importe quel autre projet Eclipse. Le projet doit uniquement inclure en plus de son propre code source la librairie Vaadin. Il peut également contenir des thèmes spécifiques au projet.

Avant de pouvoir utiliser l'application il faut bien entendu la compiler sous forme d'archive WAR avant de la déployer sur un serveur web via l'interface manager de Tomcat. En travaillant en local, le déploiement se fait automatiquement à travers les outils web de la plateforme éclipse.

7. ETUDE INITIALE DU PROJET / ANALYSE

Sur le web, il existe actuellement plusieurs plateformes qui permettent le transfert de fichiers. L'une d'entre elles, nommée « Webtransfert », hébergée par l'état du Valais, était une des principales sources d'inspiration du projet. Avant même la rédaction du cahier des charges, il m'avait été demandé d'analyser le fonctionnement de cette application.

Sur la base de cette analyse, et pour déterminer si ce projet me serait attribué, j'ai dû répondre aux questions suivantes :

- Un tel projet est-il réalisable en Java/Vaadin ?
- Ais-je les compétences suffisantes pour le réaliser ?
- Est-ce un projet assez conséquent et assez satisfaisant pour répondre aux exigences d'un travail de fin d'apprentissage ?

La caisse de compensation, placée dans le réseau de l'état du Valais n'a pas l'autorisation d'accéder aux différents sites publics de partage de fichiers tels que Dropbox ou Mediafire.

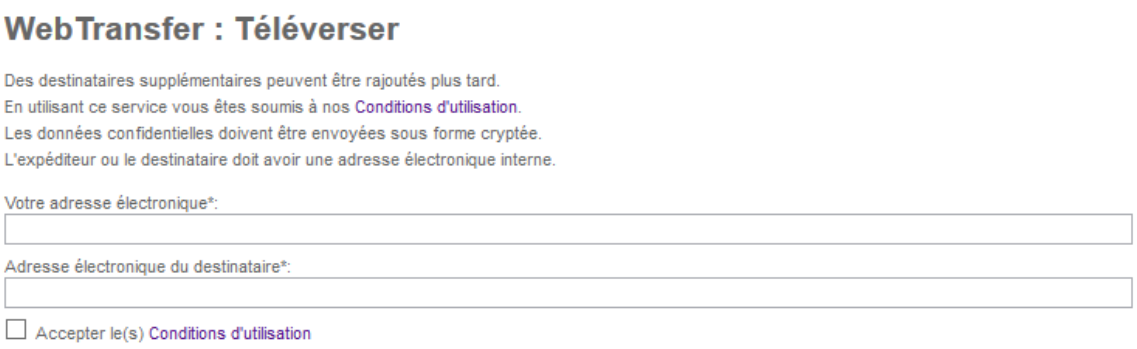
L'application « Webtransfert » actuellement utilisée pour le transfert de fichier en interne par des personnes externes travaillant pour la caisse de compensation, n'apporte pas une entière satisfaction. C'est pourquoi la caisse de compensation du Valais désire concevoir sa propre application, proposant des services similaire à « Webtransfert » tout en y apportant les modifications nécessaires, l'atout principal étant d'être propriétaire de cette application et de pouvoir ainsi la gérer pleinement.

J'ai donc pris l'initiative d'analyser l'application « Webtransfert » d'une manière plus détaillée avant la réalisation du projet pour me faire une première idée de la façon dont je voulais concevoir mon application et des conflits qui m'attendraient. Je me suis également intéressé de plus près au fonctionnement des différentes plateformes de partage de fichiers actuelles, notamment DropBox qui permet le partage de fichiers via des dossiers partagés.

Voici une conclusion brève des analyses effectuées.

7.1. WEBTRANSFERT

7.1.1. AUTORISATIONS DE TRANSFÉRER



Auteur	Destinataire		Résultat
dominique.roduit@avs.vs.ch *Interne*	dominikroduit@hotmail.com *Externe*		Pour télécharger les fichiers, vous allez recevoir un ticket de téléversement par courriel.
mathieu.berard@avs.vs.ch *Interne*	dominique.roduit@avs.vs.ch *Interne*		Pour télécharger les fichiers, vous allez recevoir un ticket de téléversement par courriel.
dominikroduit@hotmail.com *Externe*	dominique.roduit@avs.vs.ch *Interne*		dominique.roduit@avs.vs.ch will be asked if the file transfer should be made. Dès l'autorisation de transfert accordée, vous recevrez un courriel contenant un lien pour transférer les fichiers.
dominique@roduit.com *Externe*	dominikroduit@hotmail.com *Externe*		Pour la combinaison choisie, l'utilisation de ce service n'est pas autorisée. Si vous désirez acheter une copie de ce programme, veuillez consulter le site web Keelee GmbH ou utiliser la version gratuite à www.webtransfer.ch.
dominique@roduit.com *Externe*	dominique@roduit.com *Externe*		Pour la combinaison choisie, l'utilisation de ce service n'est pas autorisée. Si vous désirez acheter une copie de ce programme, veuillez consulter le site web Keelee GmbH ou utiliser la version gratuite à www.webtransfer.ch.

La demande m'a été faite de réaliser un projet similaire à WebTransfert, cependant, après avoir étudié également le concept des sites de partages actuels (Dropbox, MediaFire, …), j'ai pensé qu'il serait peut-être possible et préférable d'utiliser le concept de dossiers partagés.

Voici comment fonctionnerait le concept étudié et repensé par moi-même pour respecter les objectifs fixés dans le cahier des charges :

- ✓ Dentiste aimerait transférer un fichier à la caisse de compensation.
- ✓ Dentiste se connecte à l'application CCCVs-Transfert et saisit son adresse e-mail.
- ✓ Il n'est pas encore enregistré → Un email lui est envoyé à l'adresse qu'il a mentionnée. Ce mail contient un lien de validation.
- ✓ Dentiste confirme qu'il est le détenteur de cette adresse e-mail en cliquant simplement sur le lien reçu qui le connecte automatiquement.
- ✓ A partir de cet instant, Dentiste pourra se connecter sur l'interface à l'aide de son adresse e-mail quand il le voudra.
- ✓ Dentiste est maintenant connecté. Il se rend dans la page « contacts ». Là, il peut y ajouter une liste de contacts (noms et adresse mail).
- ✓ Dentiste ajoute info@avs.vs.ch, dominique@roduit.com et test@vs.ch.
- ✓ Dentiste se rend dans la page « gestionnaire de fichiers » dans laquelle il peut créer des dossiers auxquels il donne un nom.
- ✓ Dentiste crée un dossier nommé « Partage ».
- ✓ Dentiste peut choisir combien de temps son dossier va exister. Entre 1 et 10 jours. Il choisit 3 jours.
- ✓ Dentiste fait un clic droit sur le dossier créé et clique sur « Partager ».
- ✓ Une fenêtre s'ouvre, lui permettant de choisir auxquels de ses contacts il souhaite partager ce dossier.
- ✓ Dentiste choisit de partager le dossier avec info@avs.vs.ch. Un mail est envoyé à info@avs.vs.ch, ce mail contient un lien crypté pour accéder au dossier ainsi que des informations telle que la durée de vie du dossier.
- ✓ Dentiste accède au dossier et envoie tous les fichiers qu'il souhaite partager. Il aurait également pu le faire avant de partager le dossier, c'est égal.
- ✓ Une fois les fichiers envoyés, il peut y ajouter une description.
- ✓ De son coté, info@avs.vs.ch clique sur le lien qu'il a reçu par mail et arrive sur la page qui lui permet de télécharger les fichiers du dossier que Dentiste lui a partagé.
- ✓ Info@avs.vs.ch télécharge les fichiers, un e-mail est envoyé à Dentiste pour qu'il soit averti du téléchargement effectué.
- ✓ Trois jours s'écoulent et le dossier « Partage » créé par Dentiste ainsi que tous les fichiers contenus sont supprimés automatiquement. Un lien est envoyé à Dentiste pour l'avertir de la suppression de son dossier.

Ce concept que je trouve plus intéressant et qui ne contreviens à aucune des tâches du cahier des charges a été soumis au chef de projet Mr. Lorétan ainsi qu'au chef du service informatique Mr. Arcudi en date du 16.02.2013. Validation le 25.02.13 après discussion avec Monsieur Lorétan qui insiste également sur le fait que le processus de partage se fait également de la Caisse de compensation vers les dentistes.

Nous en avons également conclu qu'un transfert ne peut se faire que si l'adresse du destinataire ou celle de l'auteur du transfert comporte le suffixe "@avs.vs.ch".

7.2. NOMENCLATURE

La Caisse de compensation n'impose pas de règles particulières en termes de nomenclature des fichiers, dossiers, variables et tables en ce qui concerne ce projet. Cependant, lors de la validation de la base de données auprès de Mr. Lorétan, ce dernier m'a demandé de nommer mes tables d'une manière qui fasse référence au nom de l'application. Je leur ai donc assigné le préfixe « trans » pour qu'elles soient clairement identifiables.

Le nom des classes, champs et méthodes de l'application seront cependant assez explicites pour être compris par un autre développeur.

Les préfixes des libellés définissants les traductions et paramètres de la base de données seront fixés comme suit :

- APP Pour les paramètres relatifs à l'application (Nom, version)
- LAYOUT Pour les paramètres relatifs au design de l'application
- CAPTION Pour les mots/textes à traduire
- CONTENT Pour des contenus de pages à traduire
- MESS_SYS Pour la traduction des messages systèmes

7.3. JUSTIFICATION DE L'UTILISATION DE MYSQL

Mon cahier des charges stipule que mon projet fera usage du système de gestion de base de données DB2. Cependant, après de longues recherches et réflexions au sujet du cryptage, plusieurs raisons m'ont poussé à utiliser le SGBD MySQL à la place.

Ces raisons sont les suivantes :

- ✓ Mon application, sécurisée, enverra des liens par e-mail. Ces liens contiendront un paramètre crypté en SHA256. Du côté de la base de données, il faudra aller récupérer des informations en fonction du paramètre reçu dans l'URL, il faudra donc comparer la valeur cryptée d'un champ de la base de données avec la valeur reçue en paramètre dans l'URL. Il est donc indispensable d'utiliser le même algorithme de cryptage lors du hachage et lors de la comparaison qui se fera dans une condition de la requête SQL.

  Pour le faire avec DB2, plusieurs possibilités se profilaient, dont :
  - Créer mon propre algorithme de cryptage. Le même en class Java et sur DB2 comme procédure stockée.
  - Utiliser la fonction ENCRYPT de DB2 qui utilise un algorithme propre à DB2 et que je devrais donc retrouver et réécrire en Java.
  - Créer chaque fois un champ avec la valeur original et un champ pour la valeur cryptée.

  Parmi ces solutions, aucune n'a convenu, trop complexes à mettre en œuvre et surtout non formulées dans le cahier des charges. Il en existait une bien plus efficace qui ne

changeait en rien le projet mais qui nous permettait de comparer des valeurs cryptées très simplement : c'était d'utiliser une des dernières version de MySQL qui propose nativement les fonctions usuelles de cryptages (MD5, SHA1, SHA224, SHA256, SHA512, CRC32,….). Au départ, la version 5.0.36 de MySQL avait été installée car elle était disponible directement depuis YaST sur le serveur OpenSuse mais j'ai ensuite remarqué que le cryptage était seulement inclus à partir de la version 5.5 de MySQL, c'est pourquoi nous avons téléchargé et installé les packages rpm de la toute dernière version parue à ce jour (5.6.10).

- ✓ Mon application sera hébergée par un serveur dans une zone démilitarisée. Au moment de la rédaction du cahier des charges je n'ai pas envisagé ce problème mais il n'est pas possible ou en tout cas relativement peu simple d'atteindre une base de données DB2 installée sur un autre serveur en local de la caisse de compensation. Il aurait donc fallu installer mon propre serveur DB2 pour une simple base de données composée de quelques tables. Ce qui serait complètement démesuré pour un projet qui utilise une base de données de petite envergure et avec relativement peu d'enregistrements.

8. CONCEPTION

8.1. SCHÉMA DE NAVIGATION

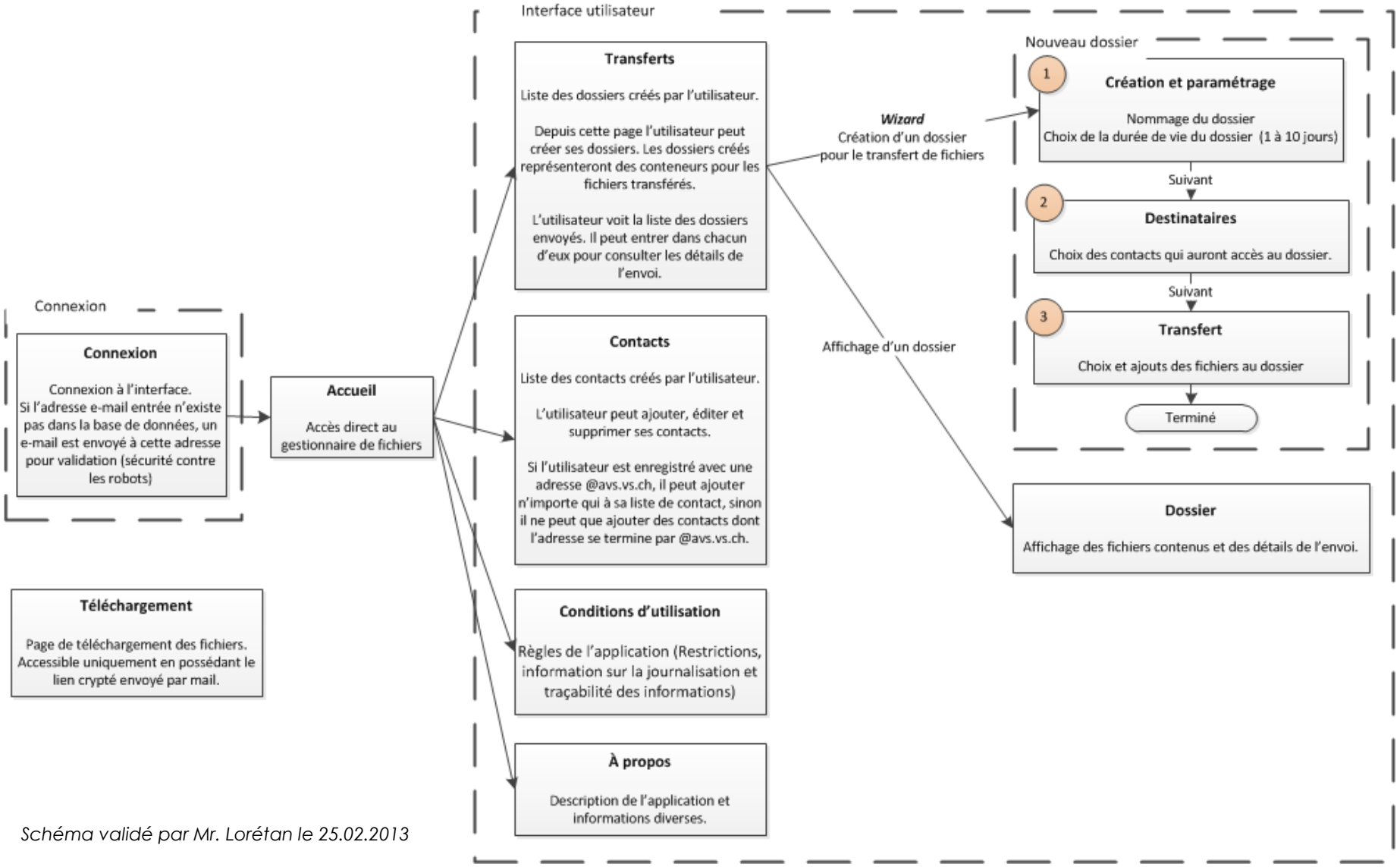


Schéma validé par Mr. Lorétan le 25.02.2013

8.2. BASE DE DONNÉES

8.2.1. SCHÉMA ENTITÉ-RELATION

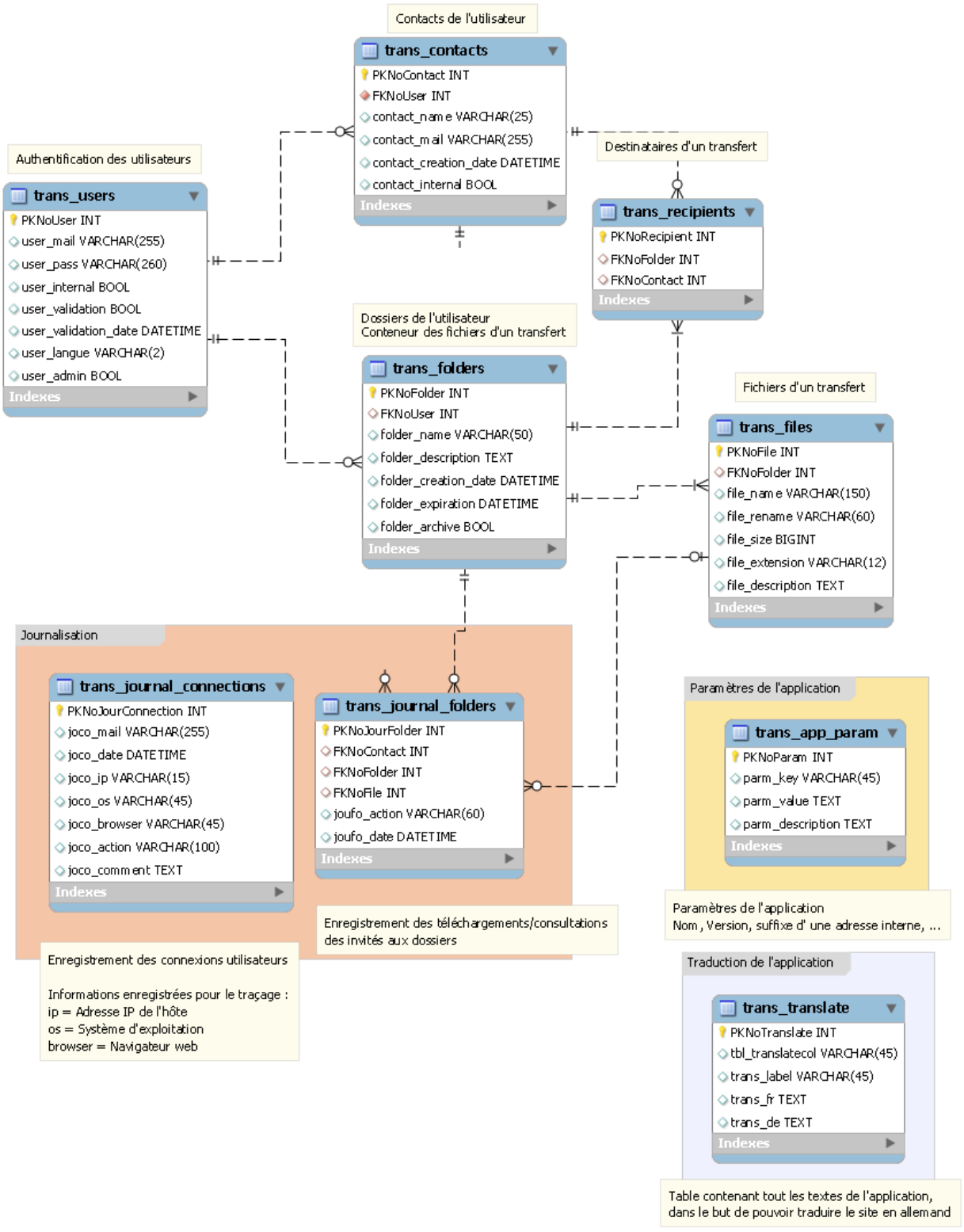


Schéma validé par Mr. Lorétan le 25.02.2013

8.2.1.1. VALIDATION

Dès la fin de sa conception, le schéma de la base de données a été soumis au responsable du projet pour sa validation. Après corrections et adaptations, il a été validé le 25.02.2013 par Mr. Yves Lorétan.

8.2.2. RÉALISATION

8.2.2.1. CRÉATION DU SCRIPT

Un script de création de la structure des tables et d'insertion des données test a été créé afin de faciliter la réalisation des tests, en particulier sur les différentes contraintes et relations.

Pour chaque table, il contient :

- La création
- Les commentaires sur la table et chacun de ses champs
- L'insertion des données de test

Les contraintes sur les clés étrangères sont créées séparément à la fin du fichier pour éviter les erreurs lors de l'importation du script SQL.

8.2.2.2. EXEMPLE

```
-- Création de la table pour les utilisateurs de l'application

CREATE TABLE IF NOT EXISTS trans_users (
  PKNoUser int(11) NOT NULL AUTO_INCREMENT,
  user_mail varchar(255) DEFAULT NULL
     COMMENT 'Adresse e-mail de l''utilisateur',
  user_pass varchar(260) DEFAULT NULL
     COMMENT 'Mot de passe de l''utilisateur (cryptage : SHA256)',
  user_internal tinyint(1) DEFAULT NULL
     COMMENT 'true : l''utilisateur travaille à la CCCVs / false : L''utilisateur viens de
     l''extérieur',
  user_validation tinyint(1) DEFAULT NULL
     COMMENT 'true : L''adresse e-mail est validée / false : La validation est en attente,
     l''utilisateur ne peut pas se connecter',
  user_validation_date datetime DEFAULT NULL
     COMMENT 'Date à laquelle l''utilisateur a validé son adresse e-mail',
  user_langue varchar(2) DEFAULT NULL
     COMMENT 'Langue de l''utilisateur',
  PRIMARY KEY (PKNoUser)
) ENGINE=InnoDB
COMMENT='Utilisateurs de l''application';

-- Insertion des données de test dans la table des utilisateurs de l'application

INSERT INTO trans_users (PKNoUser, user_mail, user_pass, user_internal, user_validation,
user_validation_date, user_langue) VALUES
(1, 'dominique@roduit.com', 'cbf61fc6215efbbf69d…', 0, 1, '2013-02-28 18:30:52', 'de'),
(2, 'dominique.roduit@avs.vs.ch', 'cbf61fc6215efbbf6…', 1, 1, '2013-02-28 18:31:34', 'fr')
```

8.2.2.3. TEST DE LA BASE DE DONNÉES

Des tests ont été exécutés à la fin de la création et ajout des données de test. Ils ont permis d'observer que tout les champs essentiels étaient présents et d'examiner chaque relations pour corriger les actions sur les références. En annexe, vous trouverez le rapport établi en conséquence.

8.3. INTERFACE WEB

8.3.1. STRUCTURE

Pour la réalisation de la partie WEB du projet qui sera développée en Vaadin, je me suis renseigné sur les différentes façons de structurer proprement ses classes Java. J'ai eu l'occasion d'analyser le code d'un projet Vaadin de la caisse de compensation et je souhaite m'inspirer de son architecture en l'adaptant à mes besoins.

Voici la base de la structure que je souhaite mettre en œuvre dans mon application :

Structure	Description
src	Dossier Racine
- common	Contient les composants personnalisés, bases de l'interface utilisateur
- component	Composants personnalisés et composant de transfert des fichiers
-upload	Package contenant les composants et interfaces utiles à l'upload
- form	Formulaires de gestion, assistants de création des dossiers, …
- global	Stockage des objets utilisés à travers l'application
- main	Package contenant l'index de l'application et les layouts d'interface
- model	Model des données de chaque table de la base de données
- modules	Modules de l'application qui forment les pages (groupes de composants)
-admin	Modules visibles uniquement par les administrateurs
- sql.query	Contient les classes qui répertorient les requêtes SQL utilisées
- toolbox	Regroupement de classes essentielles utilisées fréquemment

8.3.2. DESIGN DE L'APPLICATION

Toutes les pages principales de l'application ont été conçues avec Visio le lundi 25.02.2013. Bien qu'elles seront globalement respectées au mieux, ce ne sont que des schémas qui permettent au client (Mr. Lorétan, responsable de projet) de se représenter l'interface finale de l'application pour que toutes les corrections nécessaires se fasse rapidement avant la réalisation et non pendant. C'est pourquoi chacun des schémas présentés ci-dessous ont été validés par Monsieur Lorétan lui-même. Bien entendu, l'application réelle ne respectera pas 100% des détails graphiques et textes de ces schémas mais gardera la même structure que le client a approuvée lors de la validation. C'est aussi un bon moyen pour vous lecteur, de vous rendre compte des différences et similitudes entre les schémas conçus avant la réalisation du projet et le résultat après réalisation.

Les composants graphiques tels que les boutons et fenêtres n'auront pas le même aspect que sur ces schémas. La raison est simple : Vaadin propose ses propres composants qui ont déjà un style prédéfini. Ce style sera donc conservé dans la plupart des cas.

8.3.2.1. PAGE DE CONNEXION

8.3.2.1.1. ETUDE DU PROCESSUS

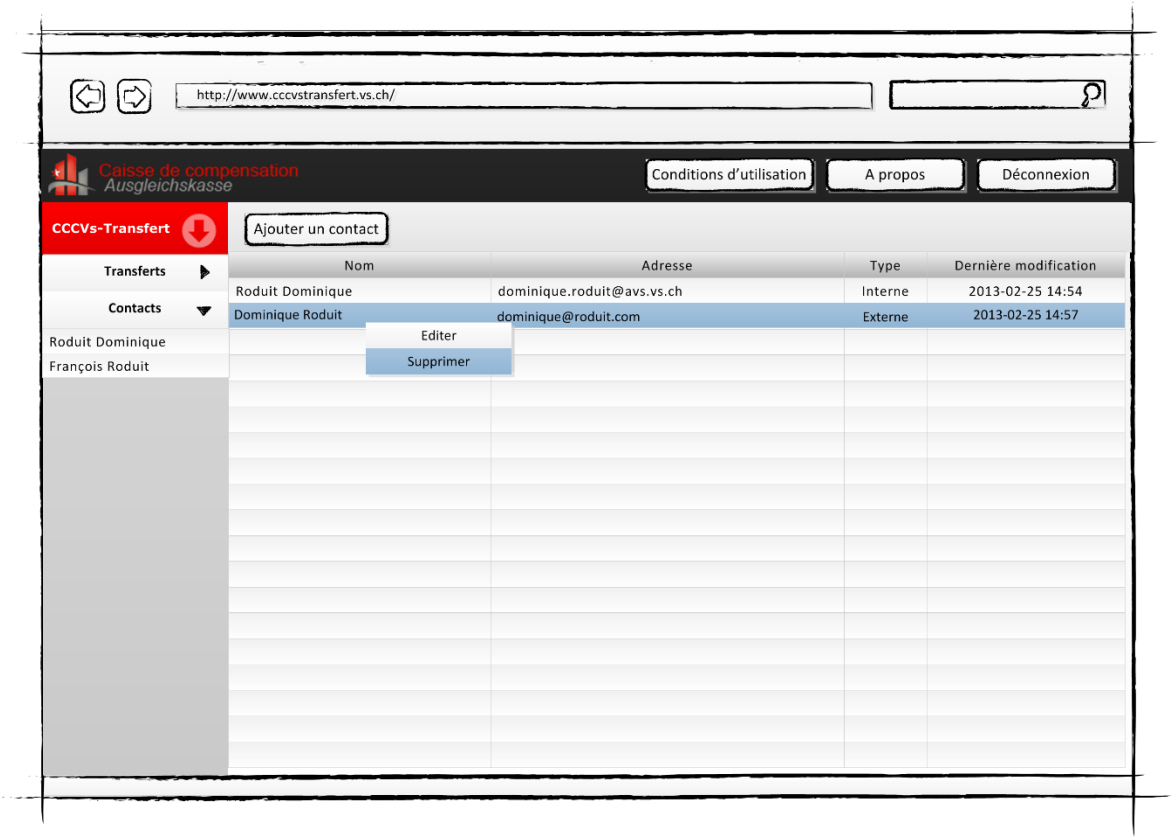
La page de connexion fait office d'inscription pour les utilisateurs qui n'ont encore jamais utilisé le service et de connexion pour ceux qui ont déjà validé leur compte. Voici comment se déroule l'inscription :

L'utilisateur arrive pour la 1e fois, il entre son adresse mail et un mot de passe quelconque, peu importe lequel. S'il n'existe pas de compte associé à cette adresse, on lui demande de confirmer le mot de passe entré en l'insérant une seconde fois. Ci fait, l'utilisateur est informé qu'un mail contenant un lien de validation lui a été transmis. Ce lien de validation fait office de sécurité, il empêche les robots de s'inscrire. L'utilisateur doit cliquer sur ce lien pour que son compte soit activé. Dès ce moment, il pourra se connecter normalement sur l'interface.

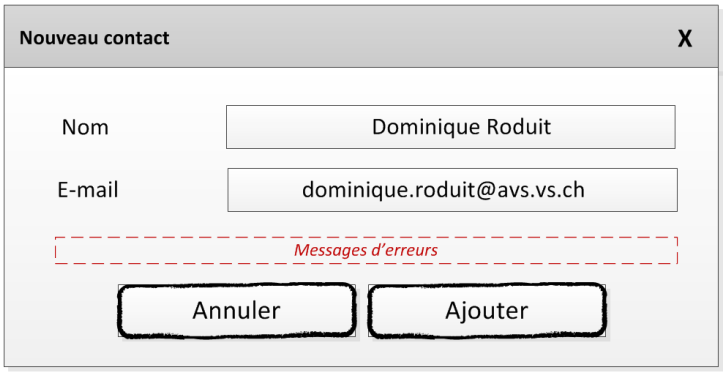
8.3.2.1.2. JOURNALISATION

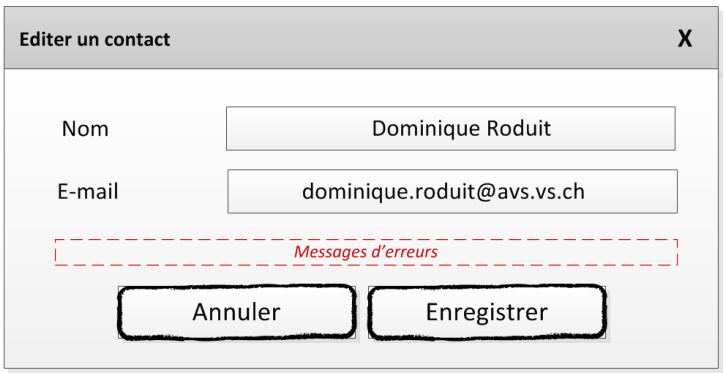
Chaque connexion, qu'elle aboutisse ou non, fera l'objet d'une insertion dans la base de données. Toutes les informations relatives à l'utilisateur sont enregistrées à chaque connexion pour nous permettre de mieux retracer d'éventuelles erreurs, des failles ou des abus d'utilisation. Les éventuelles erreurs et exceptions déclenchées par les utilisateurs lors de la connexion sont enregistrées dans la table trans_journal_connections.

8.3.2.2. PAGE « CONTACTS »



8.3.2.2.1. AJOUT ET ÉDITION DE CONTACTS

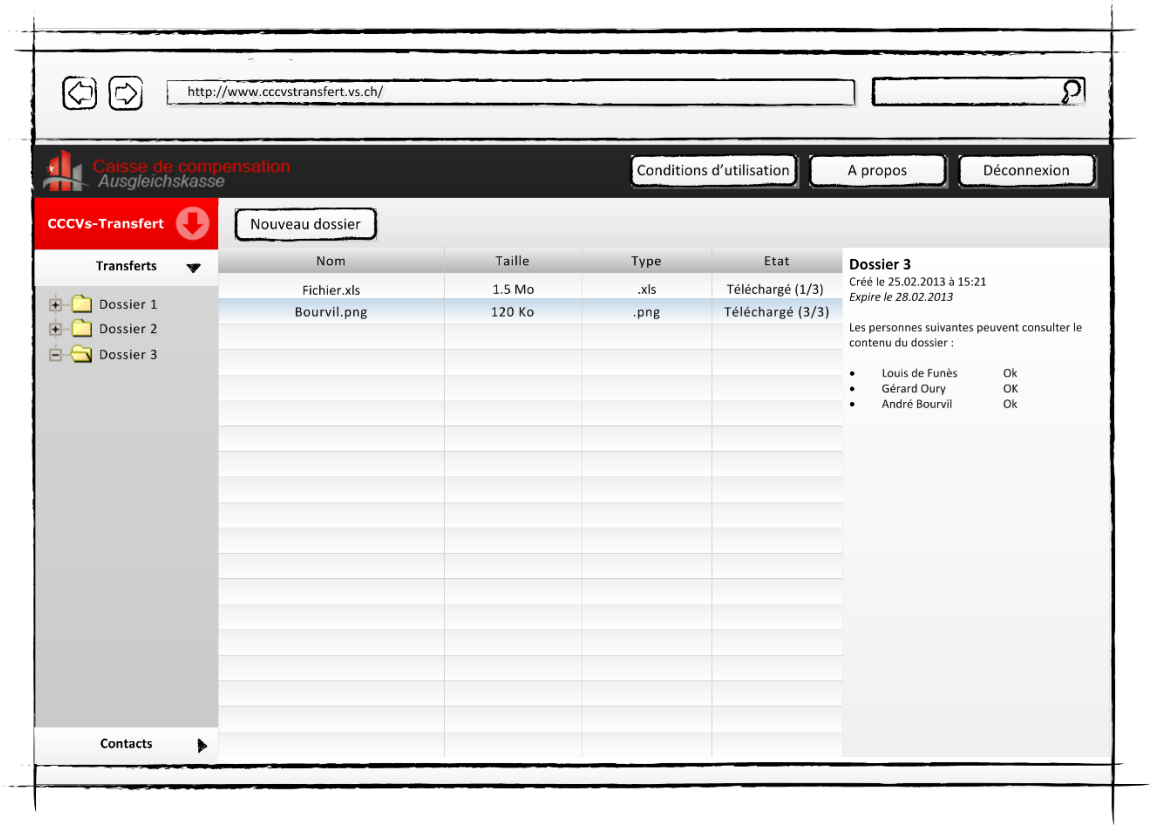




Les fenêtres ci-contre sont affichées en tant que fenêtre modales au-dessus de la liste des contacts présentée au-dessus.

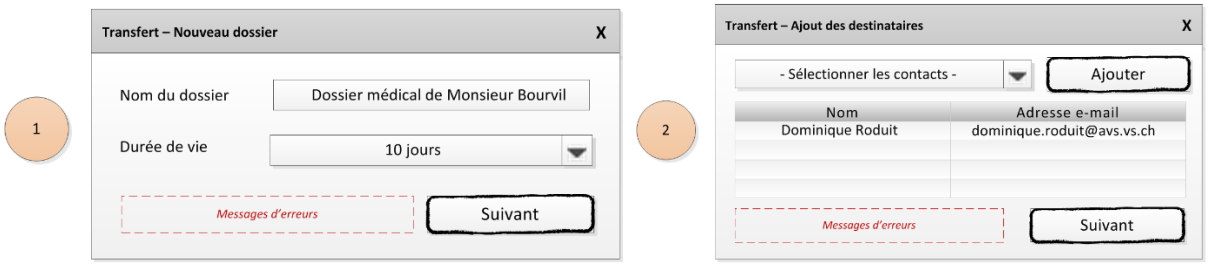
La fenêtre d'ajout régit d'un clique du bouton « Ajouter un contact » et celle d'édition, qui utilisera la même class, est affichée lorsque l'utilisateur choisi l'action « Editer » du menu contextuel qui apparaitra lors du clic droit de la souris.

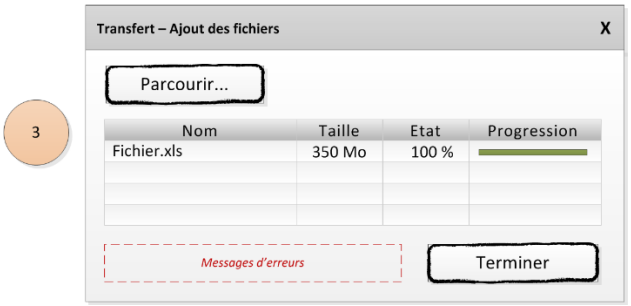
8.3.2.3. PAGE « DOSSER CONTENEUR D'UN TRANSFERTS »



8.3.2.3.1. WIZARD : DOSSIER DE TRANSFERT

Le transfert d'une personne à l'autre (envoi d'un dossier) se fera à l'aide d'une sous-fenêtre programmée sous forme d'assistant en 3 étapes.

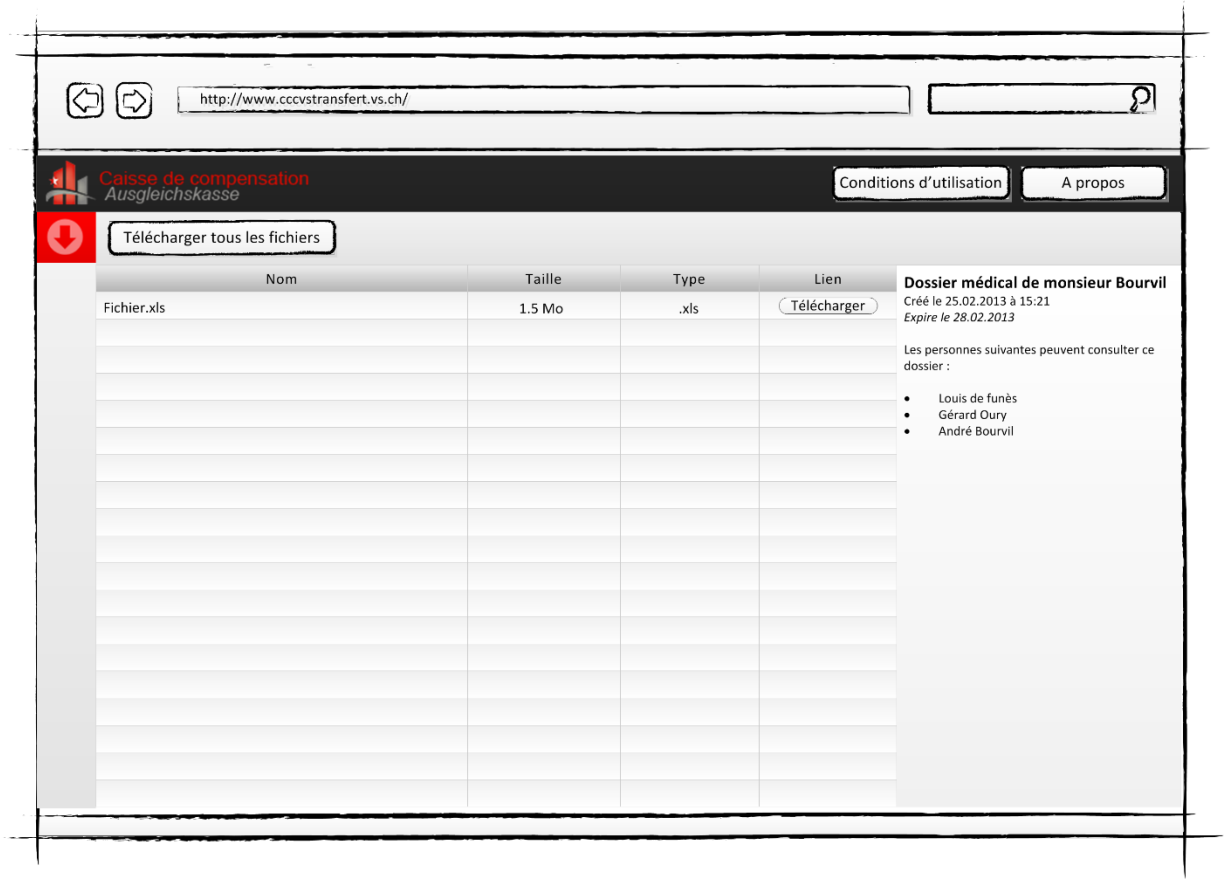




1. Saisie des informations sur le dossier
2. Sélection des destinataires
3. Sélection des fichiers à envoyer

8.3.2.4. PAGE DE TÉLÉCHARGEMENT

La page de téléchargement, accessibles via le lien crypté que les destinataires sélectionnés lors de l'étape de création du dossier recevront se présentera comme tel :



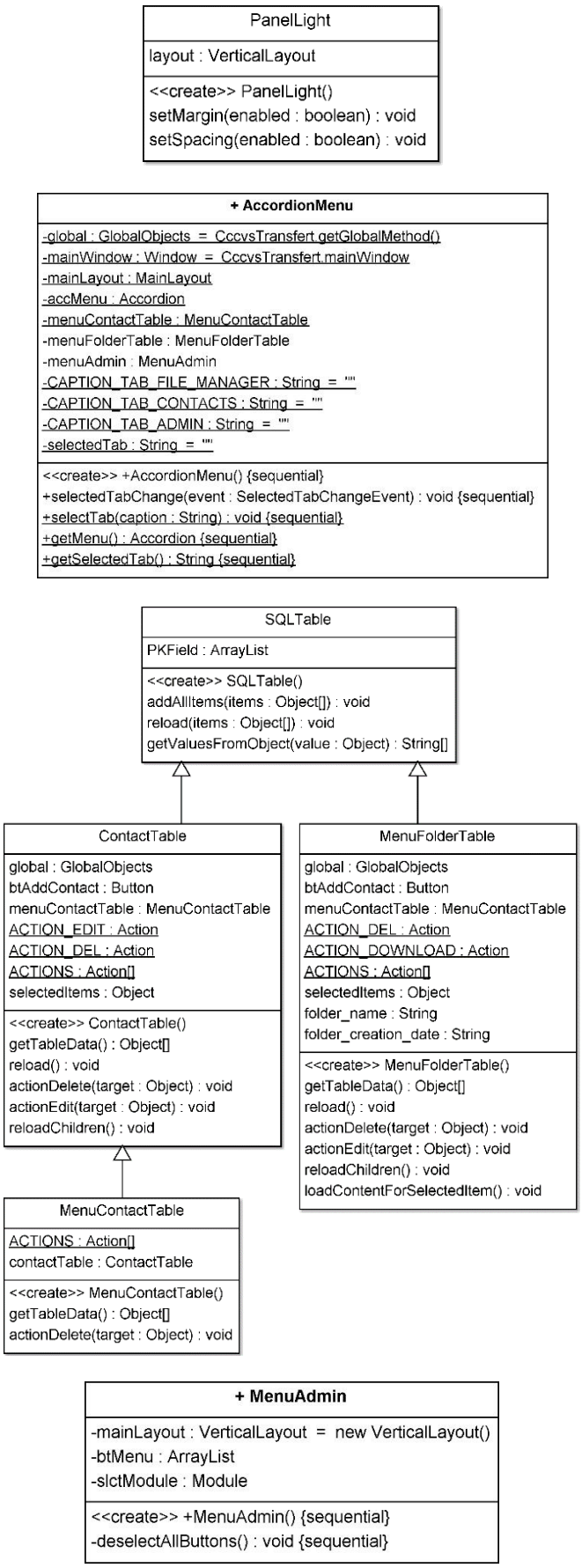
Un clic droit sur chaque fichier affichera un menu contextuel proposant le téléchargement. Chaque fois qu'un contact téléchargera un fichier ou l'archive complète du dossier, cette action sera enregistrée dans la table des journalisations pour permettra à l'auteur du transfert un suivi de son dossier.

Le simple fait qu'un contact accède à cette page est également journalisé.

8.4. SCHÉMATISATION UML DES CLASS JAVA

Les packages ne sont pas représentés car les class comportent souvent une multitude de méthodes et d'attributs et il est difficile de trouver une position adéquate pour que tout soit représenté, chaque package est donc séparé par un titre. Physiquement, les packages sont des répertoires contenant des class (fichier.java), ils servent à organiser et séparer le code de manière distincte. « common.component » indique que le package « component » est contenu dans le package « common ».

8.4.1. PACKAGE – COMMON.COMPONENT



PanelLight *(extends Panel)*
Panel dépourvu du style CSS natif de Vaadin

AccordionMenu *(extends HorizontalLayout)*

Création du composant accordéon du menu qui contient les modules ContactTable et MenuFolderTable.

SQLTable *(extends Table)*

Cette table contient des méthodes utiles pour le chargement de données depuis une base de données, elle est utilisée pour l'affichage des données d'une table de la base de données dans un tableau de l'interface utilisateur. Aternative à l'add-on Vaadin SQLContainer.

ContactTable *(extends SQLTable)*

Table pour la gestion des contacts. Cette class permet l'affichage des contacts dans un tableau. Le tableau implémente un menu, affiché sur le clic droit de la souris. Ce menu propose la suppression et l'édition des contacts. En sélection multiple dans le tableau, plusieurs contacts peuvent être supprimés en même temps, mais seule la ligne sur laquelle pointe le curseur est éditée lorsqu'il s'agit d'une édition. Lorsque l'édition est choisie, les informations sont transmises vers la class ContactForm.

MenuContactTable *(extends ContactTable)*

Table pour la gestion des contacts, affichée dans le menu latéral gauche. Cette class permet l'affichage des contacts dans un tableau. Le tableau implémente un menu, affiché sur le clic droit de la souris. Ce menu propose la suppression des contacts. En sélection multiple dans le tableau, plusieurs contacts peuvent être supprimés en même temps.

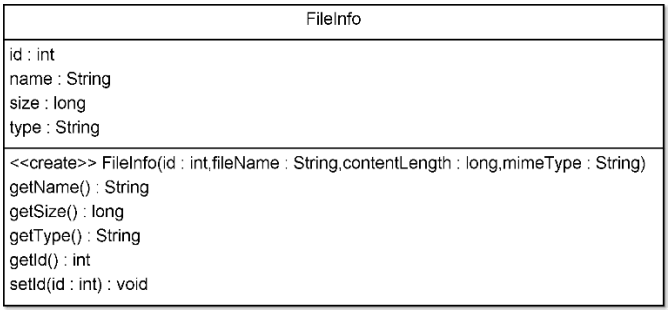
MenuFolderTable *(extends SQLTable)*

Table pour la gestion des dossiers. Cette class permet l'affichage des dossiers dans un tableau. Le tableau implémente un menu, affiché sur le clic droit de la souris. Ce menu propose la suppression des dossiers expirés et le téléchargement des dossiers toujours disponibles.

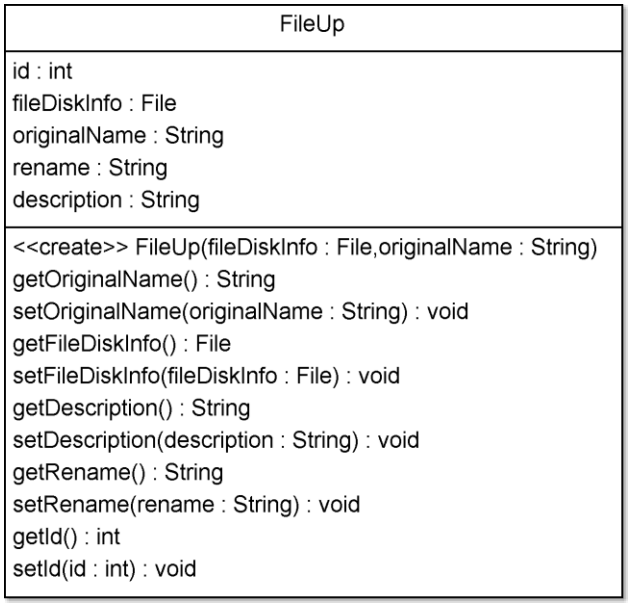
MenuAdmin

Menu affiché uniquement aux administrateurs de l'application. Il permet la gestion des utilisateurs de l'interface, la visualisation des journalisations, ...

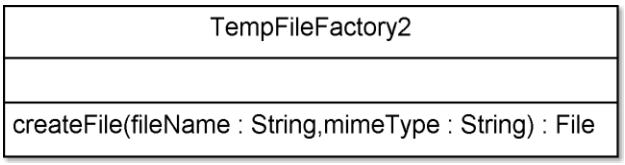
8.4.2. PACKAGE – COMMON.COMPONENT.UPLOAD



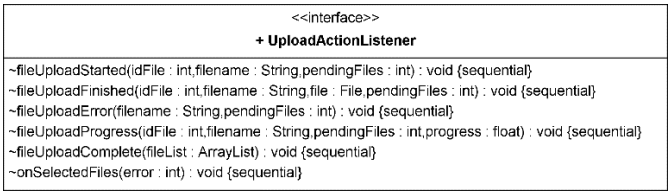
FileInfo
Class intermédiaire dont le but est de stocker des informations précises sur un fichier. On peut ensuite utiliser directement ce modèle pour récupérer tous les attributs du fichier dont on a besoin.
Utilisé exclusivement pour l'upload de fichier dans la class.



FileUp
Model d'un fichier pour l'intégration dans le tableau d'upload.
La class sert à stocker les informations sur les fichiers sélectionnés dans le but de pouvoir récupérer tous les attributs d'un fichier.



TempFileFactory2
Enregistrement temporaire des fichiers sélectionnés pour l'upload.

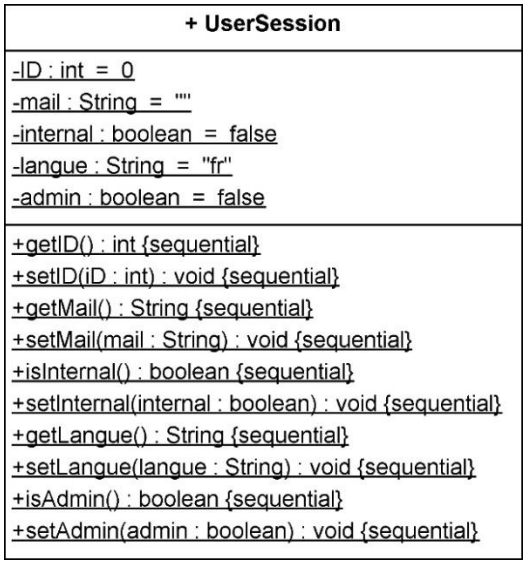


UploadActionListener
Interface contenant les évènements reçus par les fichiers lors de l'envoi. Son implémentation permet de détecter le début et la fin d'un transfert, de détecter les erreurs et de suivre la progression des fichiers.

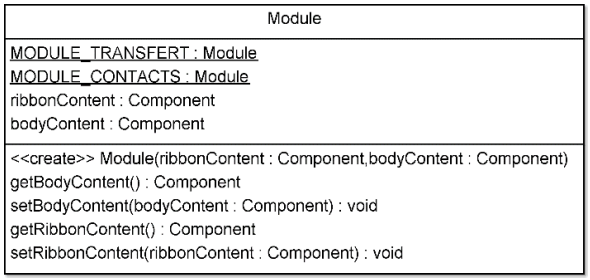
MultipleFileUpload
MultipleFileUpload est la class principale qui permet le téléchargement de fichiers multiple. Elle est basée sur la class MultiFileUpload de Vaadin qui permet le téléchargement immédiat de plusieurs fichiers en parallèles. Elle affiche également les indicateurs de progression de l'envoi des fichiers ainsi qu'une zone de drag&drop pour les navigateurs qui acceptent cette technologie.
Cette class enregistre les flux directement dans des fichiers pour limiter la consommation de mémoire.

CustomUpload
Utilisation de la classe MultipleFileUpload, récupération et implémentation des évènements de l'interface UploadActionListener. Gestion de l'affichage de la progression et de la limitation de la taille des fichiers.

8.4.3. PACKAGE – GLOBAL

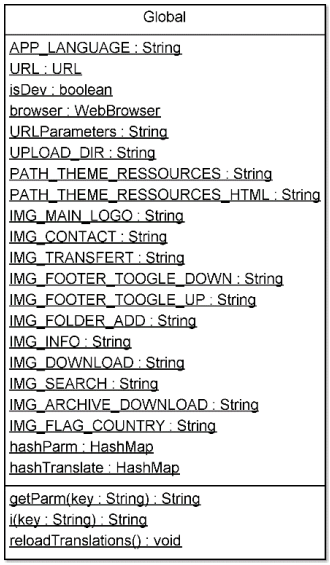


UserSession
Stockage des informations d'un utilisateur lors de sa connexion. Cette class est comparable à une variable de session en PHP. Lors du rechargement de la page, les données sont conservées grâce à un ThreadLocal. Si l'application est redémarrée, la session est détruite.

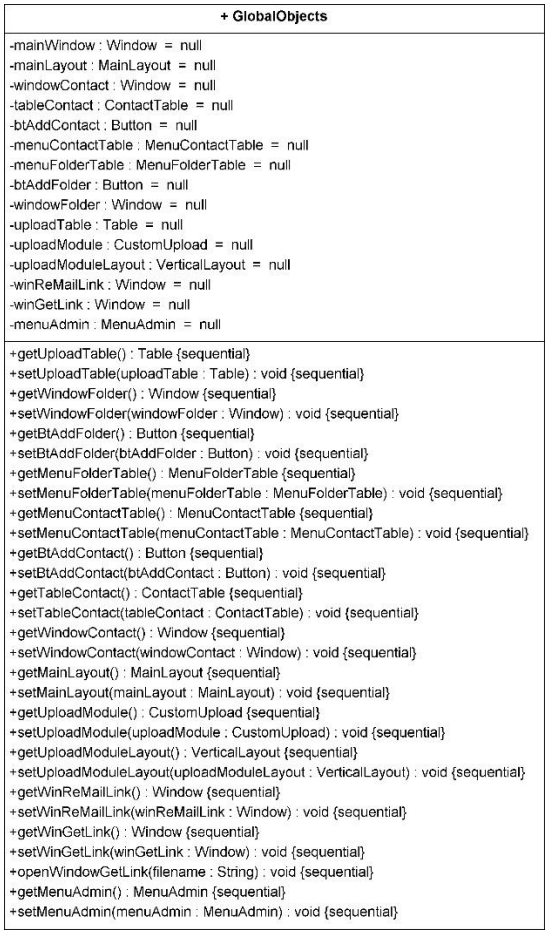


Module
Création d'un module dans le but de l'afficher sur la page. Un module se compose de deux parties :

1. Contenu du ruban = Zone haute du corps de page qui contient les boutons et les titres
2. Contenu du corps de page = Contenu principal affiché dans la partie principale de l'application



Global
Class de stockage des champs Globaux (qui doivent être accessibles pour toutes les class de l'application). Elle contient les méthodes de récupération des paramètres et des traductions de l'application. Toutes les méthodes et champs de la class sont statiques.



GlobalObjects
Class stockant tous les objets utilisés souvent, qui doivent être accessible n'importe où dans l'application. Attention cependant de ne pas utiliser un objet avant qu'il n'ait été créé et enregistré dans le champ qui lui est approprié. À la différence de la class Global, les méthodes ne sont pas statiques, la classe doit être instanciée pour que ses méthodes soient utilisables.

8.4.4. PACKAGE – FORM

+ ContactForm
-global : GlobalObjects = CccvsTransfert.getGlobalMethod() -winContact : Window = global.getWindowContact() -tblContact : ContactTable = global.getTableContact() -btAddContact : Button = global.getBtAddContact() -queries : tbl_contacts = new tbl_contacts() -mainLayout : VerticalLayout -contact : Contact -btApply : Button -form : Form -footerLayout : HorizontalLayout -buttons : HorizontalLayout -cancel : Button -contactItem : BeanItem -SubmitListener : ClickListener
<<create>> +ContactForm() {sequential} <<create>> +ContactForm(PK : String) {sequential} -doForm() : void {sequential} +getButtonApply() : Button {sequential}

ContactForm
Cette class permet l'ajout et la mise à jour des contacts via un formulaire affiché dans une sous-fenêtre. Un constructeur permet l'ajout, un autre permet la mise à jour.

+ WizFoldNew
-global : GlobalObjects = CccvsTransfert.getGlobalMethod() -winFolder : Window = global.getWindowFolder() -mainLayout : VerticalLayout -folder : Folder -btApply : Button -form : Form -footerLayout : HorizontalLayout -buttons : HorizontalLayout -cancel : Button -contactItem : BeanItem
<<create>> +WizFoldNew() {sequential} -doForm() : void {sequential}

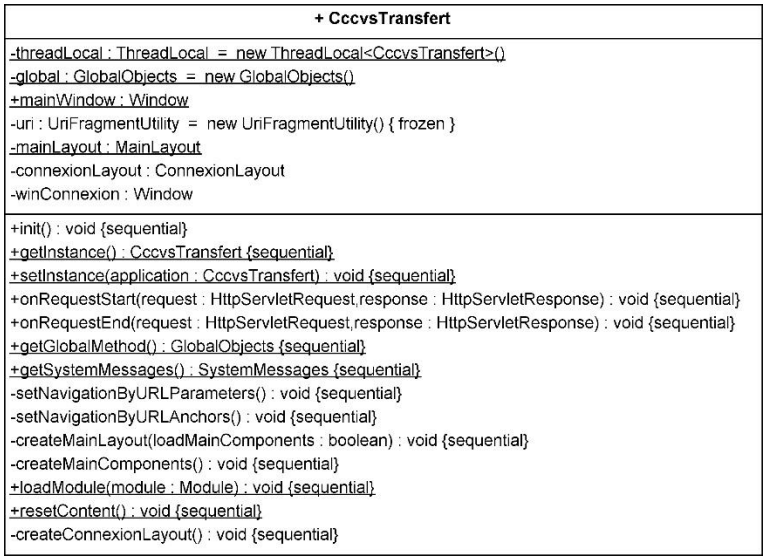
WizFoldNew
Premier Wizard de l'assistant de création d'un dossier.
Il permet la définition du nom du dossier, de sa description, et le choix de sa durée de vie.
Aucune information n'est enregistrée dans ce Wizard. Toutes les données saisies par l'utilisateur sont transmises aux vues suivantes (WizFoldDest, WizFoldUpload).

+ WizFoldDest
-global : GlobalObjects = CccvsTransfert.getGlobalMethod() -winFolder : Window = global.getWindowFolder() -mainLayout : VerticalLayout -contact : Contact -btApply : Button -form : Form -footerLayout : HorizontalLayout -buttons : HorizontalLayout -cancel : Button -container : BeanItemContainer = new BeanItemContainer<Contact>(Contact.class) -folder : Folder = null -cbxContact : ComboBox -tblRecipients : Table -slctContacts : ArrayList = new ArrayList<Integer>() -ACTION_DEL : Action
<<create>> +WizFoldDest(folder : Folder) {sequential} -doForm() : void {sequential}

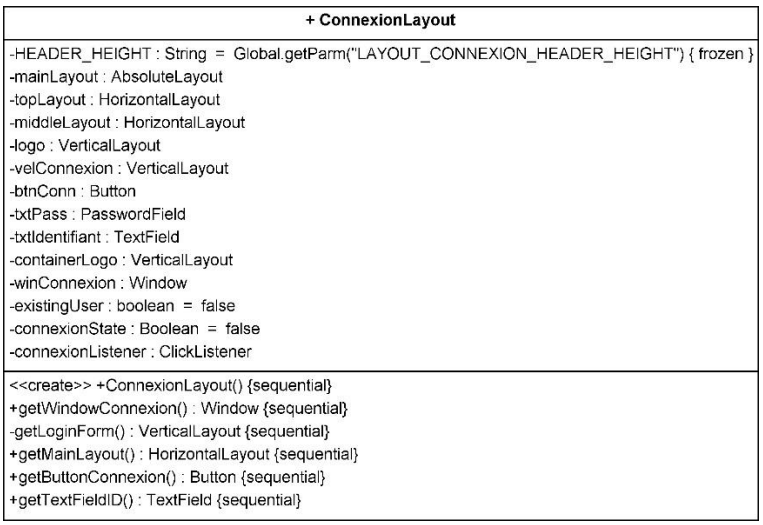
WizFolderDest
2e vue de l'assistant pour la création d'un dossier. Cette vue affiche un tableau ainsi qu'une liste déroulante contenant la liste de tous les contacts de l'utilisateur. L'utilisateur peut choisir lesquels contacts il souhaite ajouter comme destinataires.
Cette class transmet les informations sélectionnées à la vue suivante (WizFoldUpload).

WizFoldUpload
3e vue de l'assistant de création d'un dossier.
Ce Wizard permet l'upload des fichiers, le stockage dans un tableau et le renommage des fichiers envoyés.
Sur le clic du bouton "Terminer", toutes les informations sont enregistrées dans la base de données, les mails envoyés aux destinataires et l'archive contenant l'ensemble des fichiers du dossier est générée.

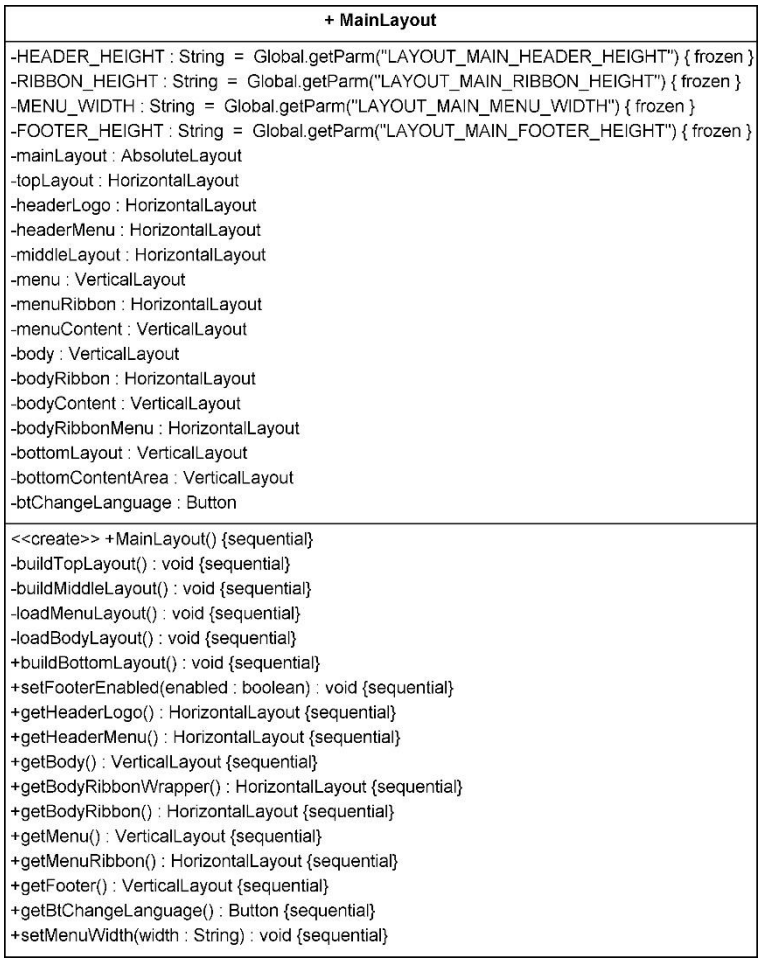
8.4.5. PACKAGE – MAIN



CccvsTransfert
Class principale de l'application qui instancie les composants racines (Fenêtres et Layouts de base). C'est cette classe qui est appelée au lancement de l'application. Elle implémente un ThreadLocal pour conserver l'état de l'application lors d'un rechargement de la page. Si l'application est redémarrée (?restartApplication), le ThreadLocal est réinitialisé.

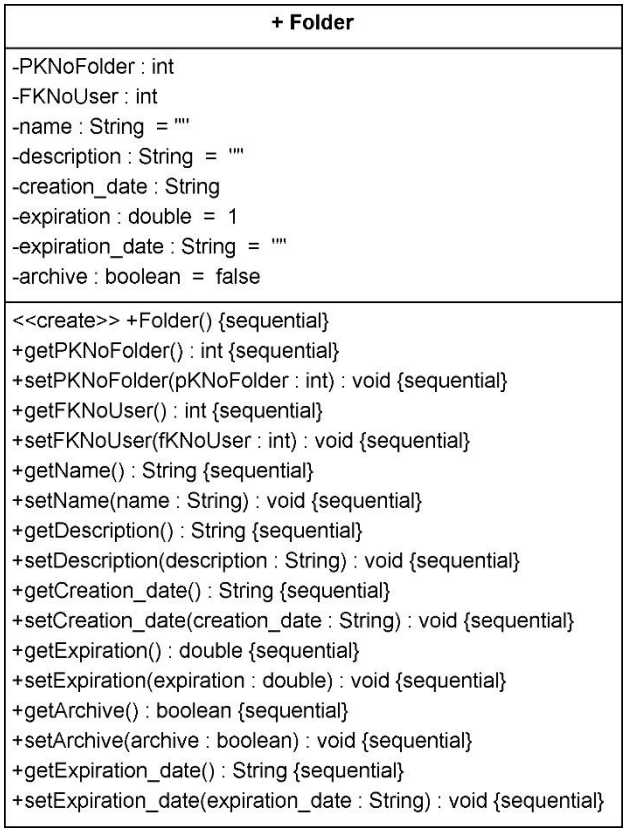


ConnexionLayout
Construction du composant de connexion à l'application.
Ce composant personnalisé est un layout qui contient le composant de connexion à l'interface utilisateur.
La class est instanciée depuis CccvsTransfert.

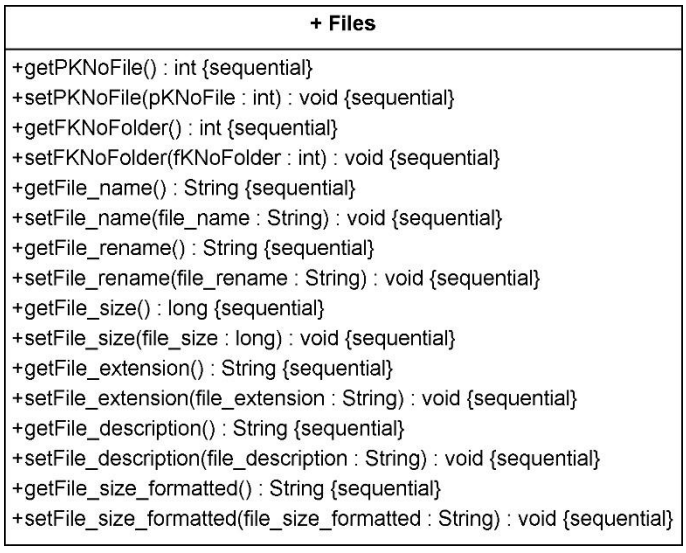


MainLayout
Ce composant personnalisé est un layout qui contient l'interface utilisateur. Il est complexe et composé de plusieurs parties distinctes (Haut, Milieu, Bas). Chaque partie est dotée d'un menu.
Cette class ne contient que les blocs et dispositions, le contenu est chargé depuis la class principale CccvsTransfert.

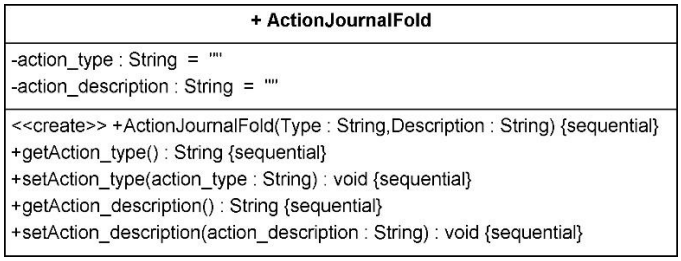
8.4.1. PACKAGE – MODEL



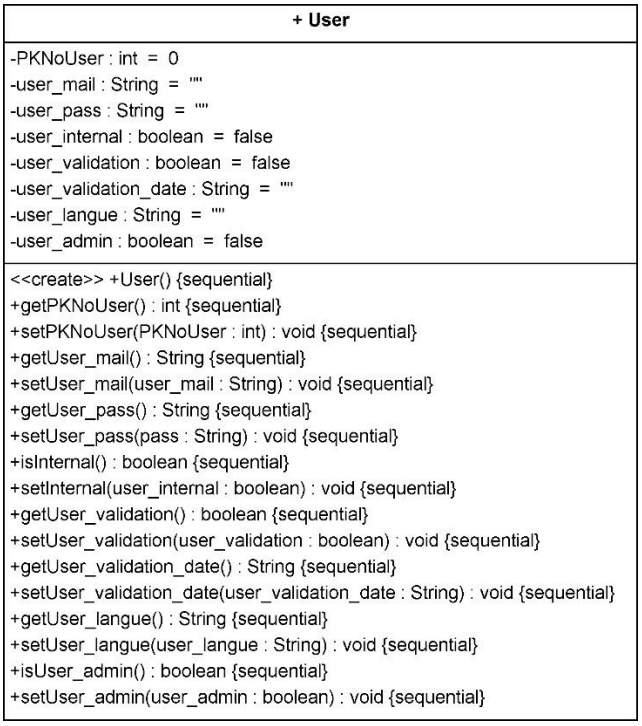
Folder
Modèle d'un dossier de transfert (Table trans_folders).
Permet le stockage des informations sur un dossier dans un objet.



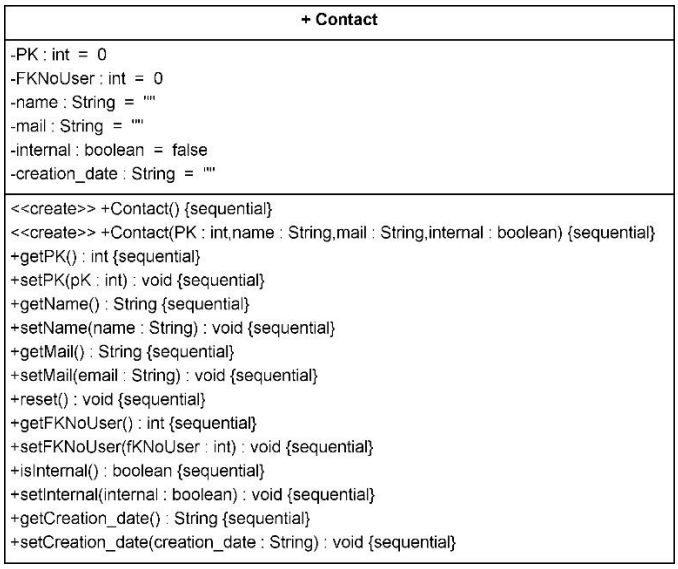
Files
Modèle d'un fichier (Table trans_files).
Permet le stockage des informations sur un fichier dans un objet.



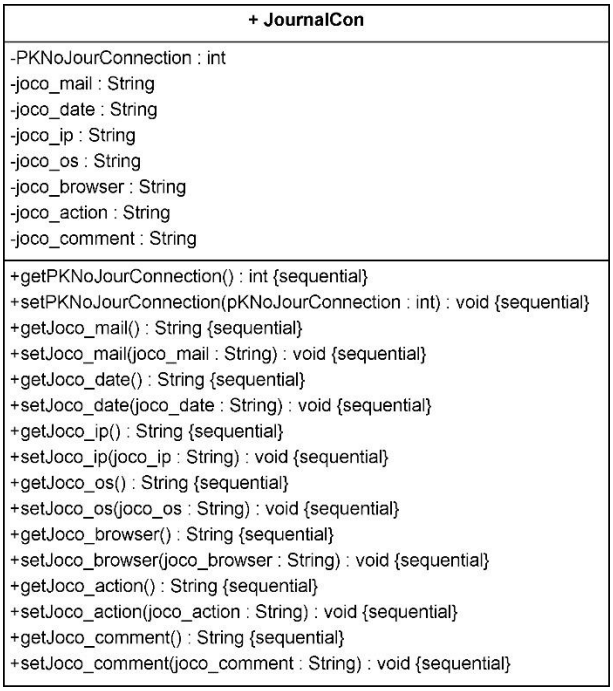
ActionJournalFold
Modèle pour le stockage d'actions journalisées sur les dossiers.
Table : *trans_journal_folders*



User
Modèle d'un utilisateur (Table trans_users. Permet le stockage des informations sur un utilisateur dans un objet.

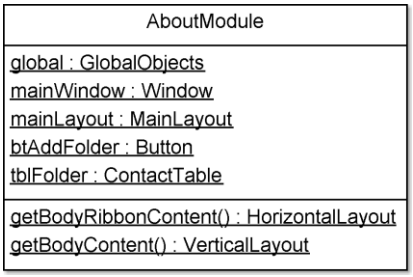


Contact
Modèle de contact. Cette class correspond aux champs de la base de données pour la table tbl_contacts.

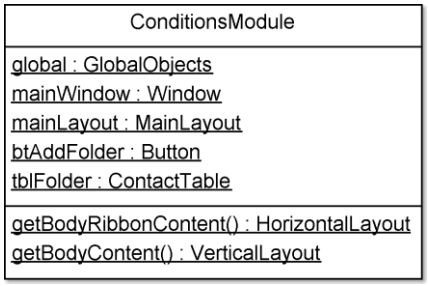


JournalCon
Modèle d'une action de connexion journalisée (Table trans_journal_connections). Permet le stockage des actions de connexion dans un objet Java.

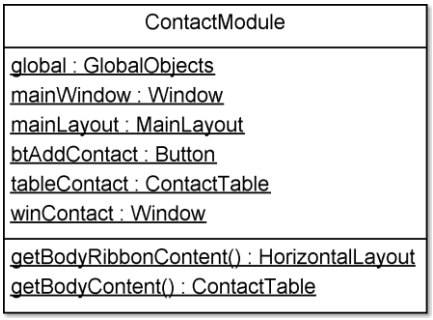
8.4.2. PACKAGE – MODULES



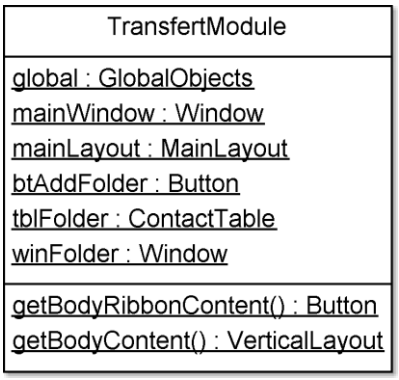
AboutModule
Page d'affichage des informations concernant l'application. La page inclut un bouton de retour au dernier module actif. Les informations sont contenues dans un bloc et illustrées par le logo de la Caisse de compensation



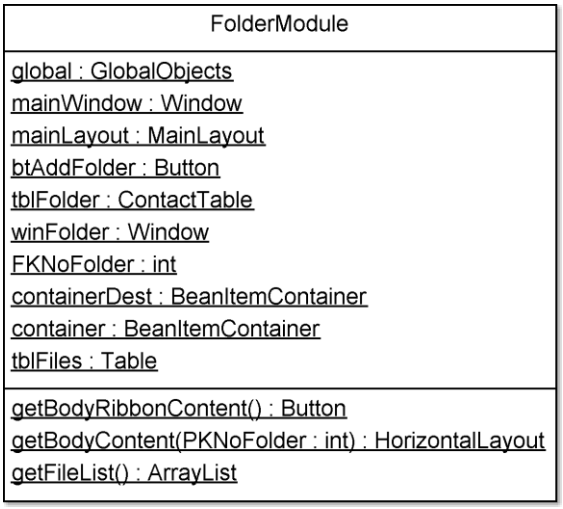
ConditionsModule
Page d'affichage des conditions d'utilisation de l'application. La page inclut un bouton de retour au dernier module actif. Les conditions sont contenues dans un bloc et illustrées par une image libre au format PNG.



ContactModule
Page de gestion des contacts.
L'utilisateur peut ajouter/éditer/supprimer des contacts.
Les actions effectuées sur cette page sont répercutées sur le tableau du menu affichant le même résultat.



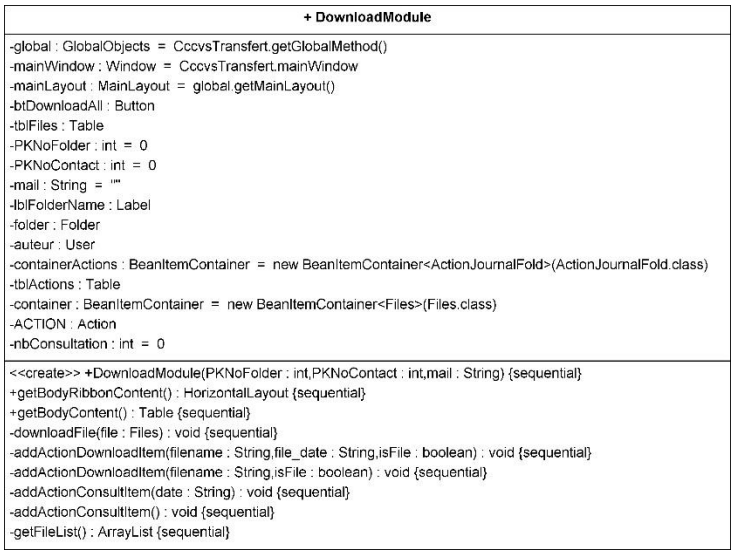
TransfertModule
Page de gestion des transferts. Elle affiche un bouton dans le ruban du menu pour permettre la création d'un nouveau dossier. La fenêtre d'assistance à la navigation est implémentée dans cette class. Lorsqu'un dossier est affiché, c'est la class FolderModule qui est implémentée.



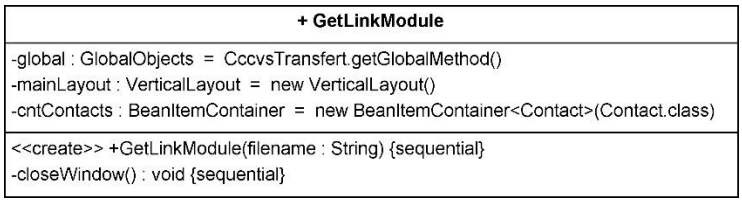
FolderModule
Page d'affichage des fichiers et informations d'un dossier. Le ruban n'est pas rechargé par cette class, le bouton d'ajout d'un dossier est donc conservé. Un panneau affiche les informations du dossier

- Création
- Expiration
- Disponibilité
- Destinataires
- Statut (archivé ou non)

Les types de fichiers sont illustrés par une icône d'extension.



DownloadModule
Page de téléchargement des fichiers d'un dossier dans lequel un ou plusieurs contacts ont été invités.



GetLinkModule
Module qui permet la récupération d'un lien de fichier ou d'un lien d'une archive de dossier. L'utilisateur peut ainsi télécharger lui-même ses propres fichiers ou partager un fichier avec n'importe qui d'autre en lui donnant uniquement l'URL vers le fichier.

+ ReMailLinkModule
-global : GlobalObjects = CccvsTransfert.getGlobalMethod() -PKNoFolder : int = 0 -mainLayout : VerticalLayout = new VerticalLayout() -cntContacts : BeanItemContainer = new BeanItemContainer<Contact>(Contact.class) -tblContacts : Table -folder : Folder
<<create>> +ReMailLinkModule(pKNoFolder : int) {sequential} -closeWindow() : void {sequential} -getContactList() : ArrayList {sequential}

ReMailLinkModule
Redistribution des e-mails pour l'accès à un dossier. Ce module peut être utile si un des contacts dit ne pas avoir reçu le mail pour X raisons, il permet de renvoyer le lien crypté. Par contre la date d'expiration est conservée et ne change pas.

8.4.3. PACKAGE – MODULES.ADMIN

+ AdminJournModule
-global : GlobalObjects = CccvsTransfert.getGlobalMethod() -mainWindow : Window = CccvsTransfert.mainWindow -mainLayout : MainLayout = global.getMainLayout() -tblActions : Table -cntActions : BeanItemContainer = new BeanItemContainer<JournalCon>(JournalCon.class) -btMenu : ArrayList -hlFilter : HorizontalLayout
+getBodyRibbonContent() : VerticalLayout {sequential}

AdminJourModule
Module disponible uniquement par les administrateurs. Il permet la visualisation rapide de toutes les actions qui ont été journalisées.

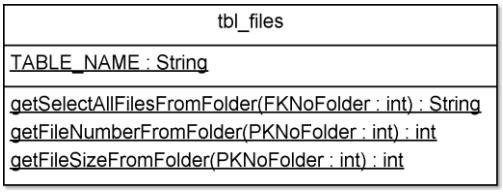
+ AdminUsersModule
-global : GlobalObjects = CccvsTransfert.getGlobalMethod() -mainWindow : Window = CccvsTransfert.mainWindow -mainLayout : MainLayout = global.getMainLayout() -tblUsers : Table -cntUsers : BeanItemContainer = new BeanItemContainer<User>(User.class) -ACTION_NO_ADMIN : Action -ACTION_ADMIN : Action -ACTION_EDIT : Action -ACTION_DELETE : Action -ACTION_VALID : Action -ACTION_INVALID : Action
+getBodyRibbonContent() : HorizontalLayout {sequential} +getBodyContent() : Table {sequential} -reloadTableUsers() : void {sequential} -getUserList() : ArrayList {sequential}

AdminUsersModule
Module disponible uniquement par les administrateurs. Il permet la visualisation et la gestion des utilisateurs de l'interface.

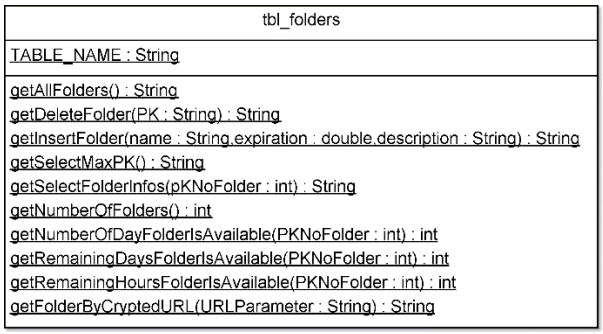
8.4.4. PACKAGE – SQL.QUERY

+ tbl_contacts
-TABLE_NAME : String = "trans_contacts" { frozen }
+getSelectAllContactQuery() : String {sequential} +getSelectByPKContactQuery(PK : String) : String {sequential} +getDeleteContactQuery(PK : String) : String {sequential} +getInsertContactQuery(name : String,email : String) : String {sequential} +getUpdateContactQuery(PK : String,name : String,email : String) : String {sequential} +getLastContact() : String {sequential} +getNumberOfContact() : int {sequential} +getContactByCryptedURL(URLParameter : String) : String {sequential} +getSelectContactsFromFolder(PKNoFolder : int) : String {sequential}

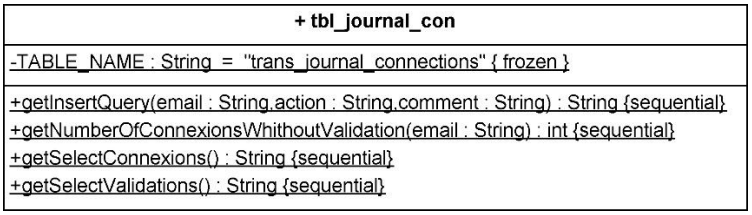
tbl_contacts
Cette class contient toutes les requêtes SQL utiles à la gestion de la table des contacts.



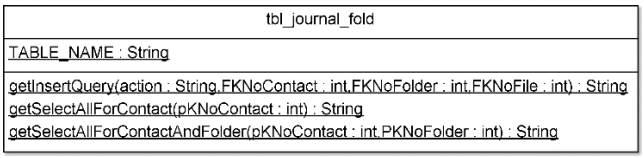
tbl_files
Contient toutes les requêtes effectuées sur la table trans_files.



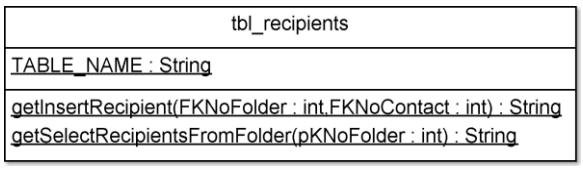
tbl_folders
Contient toutes les requêtes effectuées sur la table trans_folders.



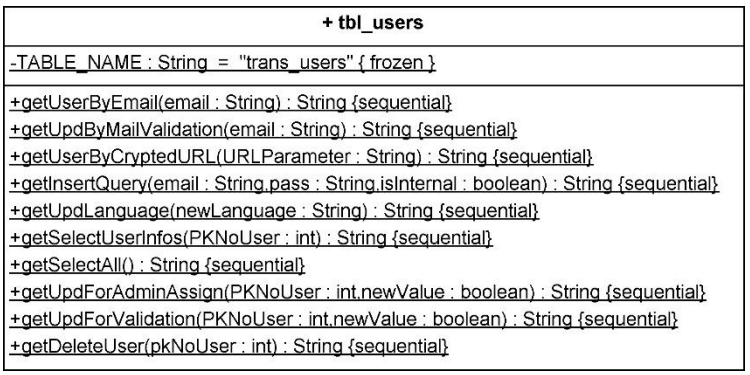
tbl_journal_con
JOURNALISATION
Contient toutes les requêtes effectuées sur la table *trans_journal_connections*.



tbl_journal_fold
JOURNALISATION
Contient toutes les requêtes effectuées sur la table *trans_journal_download*.



tbl_recipients
Contient toutes les requêtes effectuées sur la table trans_recipients.



tbl_users
Contient toutes les requêtes effectuées sur la table *trans_users*.

8.4.5. PACKAGE – TOOLBOX

Mailer
SERVER : String SUFFIXE_FORMAT : String APP_MAIL_FROM : String APP_MAIL_FROM_NAME : String
send(from : String,from_name : String,to : String,subject : String,content : String) : void send(to : String,subject : String,content : String) : void sendAccountValidationMail(to : String) : void sendDownloadLink(to : String,PKNoFolder : int,PKNoContact : int,expiration_date : String,expiration : int,description : String) : void

Mailer

Class qui répertorie différents outils relatifs à l'utilisation des adresses e-mails
Outils principaux :

- Envoi de mails en java via le protocole SMPT (requis : librairie javamail).
- Validation et contrôles de formats d'adresses

SHA256
getHashValue(str : String) : String

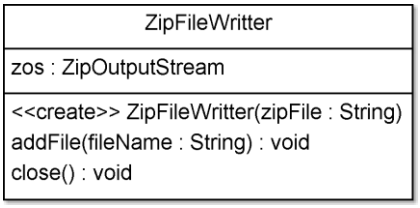
SHA256
Class de Hachage de valeurs. Cette class permet le hachage de valeur en utilisant l'algorithme de cryptage SHA-256.
Le hachage est particulièrement utilisé pour le cryptage de paramètres passés en paramètres de l'URL

Sql
serialVersionUID : long instance : Sql connection : Connection JDBC_DRIVERS : String SGBD_NAME : String SERVER : String PORT : String DB_NAME : String DB_USERNAME : String DB_PASSWORD : String rowCount : int
<<create>> Sql() getInstance() : Sql execSQLSimple(connect : Connection,SQLQuery : String) : Boolean execSQL(connectionContainer : SimpleJDBCConnectionPool,SQLQuery : String) : SQLContainer query(query : String) : ResultSet exec(query : String) : void rowCount() : int Connect() : Connection ConnectContainerTest() : SimpleJDBCConnectionPool Disconnect() : void Disconnect(connect : Connection) : void DisconectContainer(connectionContainer : SimpleJDBCConnectionPool) : void

Sql
Contient les méthodes pour la connexion et l'exécution de requêtes sur la base de données.
Les méthodes les plus importantes sont les suivantes :

- **query** : Exécution d'une requête SQL de type SELECT, que l'on peut récupérer dans un objet ResultSet.
- **exec** : Exécution d'une requête ne retournant aucun résultat, de type INSERT, UPDATE, DELETE
- **Disconnect** : Destruction de la connexion active pour la libération des ressources

Utilities
Classe regroupant les fonctions importantes utilisées globalement dans toute l'application.



ZipFileWritter
Création d'un fichier d'archive au format ZIP, compression et ajours de fichiers à l'archive.

9. AMÉLIORATIONS POSSIBLES

9.1. GESTION DES DROITS SUR LES DOSSIERS

C'est un des points sur lequel nous avons discuté le premier jour du projet mais que nous avons finalement abandonné pour cette première version de l'application. En effet, il serait très intéressant de créer une interaction, de transformer le simple transfert de fichier en une sorte de « discussion » entre l'auteur et le destinataire. Cela serait réalisable en laissant à l'auteur du transfert le choix des droits qu'il souhaite attribuer à chacun des contacts à qui il partagerait le dossier. Il y aurait 3 niveaux de droits possibles :

1. Lecture seule — Le destinataire n'a que la permission de télécharger les fichiers
2. Lecture-écriture — Le destinataire peut télécharger et déposer de nouveaux fichiers
3. Tous les droits — Le destinataire a les mêmes droits que l'auteur.

A ce moment, il y aurait beaucoup d'autres contraintes à mettre en œuvre. Notamment l'évolution continue du dossier. Une fois le dossier créé il devrait rester modifiable à souhait. Ce serait une solution intéressante et qui pourrait se révéler très efficace selon moi mais elle représente probablement une masse de travail beaucoup plus conséquente et irréalisable en si peu de jours.

9.2. AMÉLIORATION DE LA JOURNALISATION

La journalisation prévue dans le cahier des charges ne prend pas en compte certaines actions telles que la simple visualisation d'un dossier par un utilisateur agréé. Il est intéressant pour l'auteur d'un transfert de savoir si les fichiers qu'il a transmis sont au moins consultés, ce serait donc un plus d'ajouter cette possibilité. **EFFECTUÉ**

9.3. MAIL DÉTAILLÉ SUR LES ACTIONS UTILISATEURS

J'ai pensé qu'il serait bien d'envoyer un mail à l'auteur d'un dossier lorsque celui-ci expire. Ce mail afficherait un résumé des actions effectuées par les utilisateurs en intégrant un tableau qui présenterait quel utilisateur à téléchargé quel fichier, lequel n'a pas accédé, etc… **EFFECTUÉ**

10. Liens utiles et références

10.1. Liens utiles

Site web de la CCCVs	http://www.avs.vs.ch/
Site web du framework Vaadin	http://vaadin.com/

10.2. Sources & Références

10.2.1. Gabarit Visio pour la schématisation web

Visio Stencils for Information Architects	http://bit.ly/MC8DT
Visio Wireframe Stencil	http://bit.ly/wj1ip
Visio sketch shapes	http://bit.ly/XWpTMD
Templates & Stencils for Visio & Omnigraffle	http://bit.ly/3hfRNu

10.2.2. Design de l'application

Icônes (application et documentations)	http://www.iconfinder.com/

10.2.3. Navigation

ThreadLocal	http://bit.ly/125W6sl http://bit.ly/XpAWBz
Ancres (URIFragmentUtility)	http://bit.ly/YQI2OD
Paramètres URL (URIHandler)	http://bit.ly/12kYo26

10.2.4. Upload de fichier

EasyUploads (Add-on Vaadin)	http://bit.ly/14Nb9Wm
Google project – Mise à jour de l'add-on (Serveur SVN)	http://bit.ly/Yo8OlN
Configurations tomcat	http://bit.ly/11yOr6i

10.2.5. Certificat SSL

StartCom, Autorité de certification SSL gratuite	https://cert.startcom.org/
Génération et configuration d'un certificat SSL	http://bit.ly/gz8g5

10.2.6. Cryptage en Java

SHA-256	http://bit.ly/bWzO1I

10.2.7. Envoi d'e-mails

Utilisation de la librairie JavaMail	http://bit.ly/WE5PTl
Serveur SMTP gratuit pour Windows (pour des pré-tests)	http://bit.ly/X1S4ud
Validation d'adresses E-mail	http://bit.ly/2DlowW
SMTP, via port 25 ou 587 ?	http://bit.ly/Z5pJEL

10.2.8. Suppression automatique des fichiers : Crontab

Création d'une archive JAR exécutable	http://bit.ly/XfOJbc

11. CONCLUSION

11.1. BILAN PERSONNEL

Avant le commencement du projet j'avais une grande appréhension quant à la programmation en Vaadin. Bien qu'étant l'un des outils de développement principal de la Caisse de compensation, outre les quelques heures accomplies dans le cadre d'une formation avancée offerte par l'entreprise je n'en avais jamais pratiqué. Certains aspects m'ont paru plus compliqués qu'ils ne l'étaient manifestement, d'autres m'ont semblé simples et se sont révélés largement plus ardus. Grand amateur des nouvelles technologies du web que je suis, j'avais pourtant toujours répugné les librairies qui selon moi « faisaient tout à notre place ». J'avais tendance à toujours vouloir réinventer ce qui existait déjà, dans le but de progresser sans doute, mais ce projet m'a permis de découvrir d'autres façons de procéder, d'autres manières de programmer, j'ai eu la chance d'assimiler un « langage » (Vaadin) et d'en approfondir un deuxième (Java). Je reste persuadé qu'en dehors du contexte d'évaluation, ce travail fut aussi pour moi un apport important de connaissances diverses qui me seront vraisemblablement très utiles pour la suite de ma formation. J'ai conscience qu'il pourra faire office de « carte de visite » professionnelle à l'avenir et c'est pourquoi j'y ai mis toute ma volonté, en espérant bien sûr que le résultat soit prometteur et plaise aux différents partis impliqués dans la réalisation d'une telle épreuve.

11.2. BILAN TECHNIQUE

Malgré les nombreux petits problèmes rencontrés tout au long de la réalisation et la perte d'un jour complet de documentation - que je me suis empressé de réécrire - , le projet s'est très bien déroulé. La répartition du temps a certes été bien mal estimée. En analysant ma planification réelle, j'en conclu que les tâches auraient dû être beaucoup plus en parallèles car il est rare que j'aie eu à travailler sur une fonctionnalité sans qu'une autre en sois impactée.

Cependant, la partie 1 du cahier des charges tel qu'il a été signé est complétement réalisée. Le temps imparti m'a même permit d'exécuter la partie optionnelle (2e partie) et d'y ajouter plusieurs autres fonctionnalités personnelles (Paramétrage de l'application, Traduction en allemand, Limitation des extensions de fichier, Page de gestion des contacts, Affichage de l'historique des actions journalisées pour un dossier, etc…).

12. REMERCIEMENTS

Je souhaite remercier tout particulièrement …

François Roduit
Pour ses conseils et corrections orthographiques avisées.

Stefen Ewert
Pour son aide au développement de l'add-on Vaadin EasyUploads. - *http://www.c-onservices.de/*

Yves Lorétan
Pour m'avoir permis de m'épanouir dans ce projet en laissant libre cours à mon imagination en ce qui concerne tout le processus de transfert en intégrant la notion de répertoires.

Philippe Arcudi & Adrien Donnet-Monay
Pour la mise en place de toute l'infrastructure réseau-système et la mise à disposition des logiciels nécessaires.

Nicola Iannace
Pour son aide au débogage des quelques problèmes rencontrés au niveau du Vaadin.

Mathieu Bérard
Pour son aide à l'exportation du projet JAR de suppression automatique des fichiers.

Thibault Schönmann
Pour son aide à la compréhension des erreurs Java que je peinais à résoudre

Laure Balet, Eric Olivier ainsi que tous les collaborateurs du service informatique
Pour leurs conseils et corrections des documentations

13. ANNEXES

- ✓ Journal de bord
- ✓ Procès-verbal
- ✓ Rapports de tests
- ✓ Documentation technique
- ✓ Manuel d'utilisation
- ✓ JavaDoc
- ✓ Documents de préparation de l'infrastructure
- ✓ Codes sources personnel

14. DATE ET SIGNATURE

Date	Signature
Sion, le 27.03.2013	Dominique Roduit

CCCVs-Transfert

Journal de bord

Caisse de compensation du canton du Valais
Ausgleichskasse Wallis

Service informatique

Dominique Roduit

27/03/2013

dominique.roduit@avs.vs.ch

Lundi, 25 Février 2013

Février / Mars

Lundi 25 — Jeudi 28 — Vendredi 01 — Lundi 04 — Jeudi 07 — Vendredi 08 — Lundi 11 — Jeudi 14 — Vendredi 15 — Lundi 18 — Jeudi 21 — Vendredi 22 — Lundi 25 — Mardi 26 — Mercredi 27

Périodes		Déroulement	
		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	Discussion avec le responsable à propos de ma proposition pour la modification du processus de partage comme décris dans l'étude préalable de la documentation du projet.	La proposition est acceptée par le responsable, par contre il ajoute que je ne dois pas oublier que le processus doit être valable de l'interne à l'externe (car je ne le précisais pas dans mon étude préalable)
08:30	09:30	Conception du schéma de navigation à l'aide du logiciel Microsoft Office Visio 2010.	Première validation du schéma au responsable. Le schéma n'est pas validé, il n'est pas assez clair et grossit considérablement la masse de travail. Ce n'est pas prévu dans le cahier des charges.
09:30	10:50	Corrections du schéma de navigation et validation au chef de projet. Insertion à la documentation du projet.	Les derniers détails de tournures de phrases sont corrigés et le schéma validé. Je peux commencer le schéma de base de données
10:50	12:00	Conception du schéma de base de données avec MySQL Workbench.	Le schéma est terminé, il y a certainement quelques détails qui peuvent être discutables, je ferai valider ce schéma à Mr. Lorétan dès son retour. En attendant, je vais commencer la conception de l'interface Web.
12:00	13:00	Première ébauche d'une page de l'interface utilisateur avec Visio 2010.	
13:00	14:00	Pause de midi	
14:00	14:30	Validation du schéma de base de données et de la première ébauche de l'interface utilisateur auprès de Mr. Lorétan. Corrections du schéma de base de données.	La base de données comporte quelques champs inutiles et il faut en rajouter quelques-uns qui manquent. Mr. Lorétan me dit qu'il serait bien de prévoir aussi la traduction FR-DE de l'application. Je rajoute donc une table pour cela.
14:30	16:00	Conception de toutes les pages de l'interface utilisateur et validation auprès du responsable.	Le schéma de base de données est validé, il y a juste encore quelques modifications à apporter au niveau du nommage : modifier les préfixes des tables « tbl_ » par « trans_ » et ajouter une table pour la journalisation des téléchargements mais pas besoin de lui remontrer a dit Mr. Lorétan. Le design de l'interface web est bon et validé.
16:00	17:00	Dernières modifications du schéma de la base de données et conception du design de l'interface de la page pour le téléchargement des fichiers que j'avais oubliés de faire... Génération du script de base de données avec MySQL Workbench, adaptation du script, ajout de commentaires sur chaque champ et chaque table, insertions de données de test. Importation du script pour effectuer les tests.	

Problèmes rencontrés & résolution

En dehors des problèmes de logiciels qui ne marchaient pas comme je le voulais, je n'ai rencontré aucun problème relatif au projet aujourd'hui.

Objectifs pour le prochain jour

Je suis en avance de 4h sur la planification, soit une demi-journée, ce qui est très peu car je sais qu'on a vite tendance à prendre du retard par la suite, j'ai pu l'expérimenter dans les quelques projets de préparations que nous avons eu l'opportunité de faire à l'école. La prochaine journée s'annonce chargée : il me faudra réaliser la base de données, construire sa structure, la charger de données pour la tester, éventuellement la refaire valider par Mr. Lorétan. J'envisage également de commencer au moins la création du design mais il sera très difficile d'arriver jusque-là puisque j'aurai la visite des experts et de mon titulaire de l'EMVs pour l'évaluation de stage. Mon cahier des charges sera respecté mais plusieurs fonctionnalités n'y figurant pas ont été ajoutées aujourd'hui durant l'étude détaillée du projet, tout au long de cette étude détaillée du projet et je vais donc devoir mettre les bouchées doubles pour réussir à toutes les terminer. Je vais néanmoins me concentrer avant tout sur le cahier des charges pour éviter de me piéger.

Tâches réalisées

1 Conception
- 1.1 Schéma de navigation — 100%
- 1.2 Base de données — 100%
- 1.3 Interface web — 100%

2 Réalisation

2.1 Base de données
- 2.1.1 Création — 90%

Jeudi, 28 Février 2013

Février / Mars

Lundi 25
Jeudi 28
Vendredi 01
Lundi 04
Jeudi 07
Vendredi 08
Lundi 11
Jeudi 14
Vendredi 15
Lundi 18
Jeudi 21
Vendredi 22
Lundi 25
Mardi 26
Mercredi 27

Périodes		Déroulement	
		Tâches effectuées	**Remarques sur les tâches effectuées**
08:00	09:00	Tests de la base de données. Vérification de la création de tous les champs, de leur types, des clé primaires, des relations et actions sur les clés étrangères.	Les tests m'ont permis de me rendre compte des erreurs que j'avais faites. Il y en avait peu, certes, mais les actions sur les différentes clé étrangère (ON DELETE CASCADE) qu'il manquait sont importantes. Tout a été corrigé à mesure.
09:00	10:00	Séance avec Monsieur Schönmann, titulaire de l'école des métiers pour la qualification de stage.	Ce temps ne fait pas partie de mon TPI, je travaillerai donc une petite heure en plus ce soir pour compenser ce manque.
10:00	11:00	Séance avec les deux experts responsables du suivi de mon projet.	La séance m'a apporté beaucoup de réponses aux questions que je me posais, j'ai pu montrer aux experts ma planification, où je me situais. Elle m'a permis de me rendre compte également des petites choses qu'il manquait dans mon analyse technique, particulièrement les schémas de class qu'il faut que je fasse.
11:00	12:00	Rédaction du Procès-verbal de la séance avec les experts et prise de connaissance des documents relatifs aux exigences du TPI fournis par les experts.	Envoi du procès-verbal à Mr. Lorétan et Mr. Arcudi, proposition des deux heures possibles pour la défense du projet à Mr. Lorétan (par mail puisqu'il n'est pas présent).
12:00	12:30	Correction du schéma de navigation et envoi de la planification initiale par mail aux experts	Je me suis rendu compte en présentant mon schéma de navigation aux experts qu'il manquait la page « Téléchargement » dans le schéma de navigation. Je l'ai ajoutée
12:30	13:00	Design de l'application, création du projet Vaadin et de l'architecture	Je commence à préparer la structure du projet, donc les packages et les class qui seront très logiquement utilisées. Je créé un nouveau thème Vaadin personnalisé, basé sur un thème natif.
13:00	14:00	Pause de midi	
14:00	15:00	Création du script de connexion à la base de données, de quelques méthodes utiles pour exécuter les requêtes.	Test de la connexion en local puis sur le serveur distant. La connexion est établie, j'ai créé une méthode statique très utile pour l'exécution des requêtes, elle peut être appelée depuis n'importe où et n'importe quand.
15:00	15:40	Construction du design du layout pour la connexion	Création du script de vérification de l'adresse, de l'existence de l'utilisateur, … Préparation des cas d'erreur.
15:40	16:00	Apéro de départ d'un collaborateur	Apéro pour le départ de Julien Pitteloud qui nous quitte après 4 ans en tant que collaborateur au service informatique. Je reste un peu plus longtemps à la fin pour

			rattraper ce temps qui ne fait pas partie du TPI.
16:00	17:00	Création d'une méthode dans la class Global pour la récupération de paramètres et de traductions dans la base de données.	Il suffit de faire Global.getParm(nom_parametre) et la fonction nous retourne la valeur. Pour obtenir un mot traduit : Global.i(le_mot). J'ai utilisé des HashMap pour le faire, c'est selon moi une solution valable puisqu'elle nous permet de charger une seule fois le tableau de valeur et de l'utiliser ensuite dans toute l'application. Ça évite les allers et retours entre l'application et la base de données.

Problèmes rencontrés & résolution

Problème de centrage des éléments pour la partie de création du design. Je n'ai pas réussi à trouver la solution pour l'instant avec le composant que j'ai choisi (Label dans un VerticalLayout) mais je pense que je réussirai à la trouver demain en essayant une autre alternative.

Objectifs pour le prochain jour

Je ne pensais pas arriver aussi loin ce deuxième jour. Malgré les deux séances du matin qui m'ont pris deux heures, je suis toujours très en avance sur ma planification. J'ai hésité à la modifier un peu ce matin avant la venue des experts mais je ne pense pas que ce soit une bonne chose maintenant que j'ai commencé sur ce chemin. Je sens arriver que certaines tâches me prendront bien plus de temps que ce que j'ai planifié et je préfère donc garder le plus d'avance possible tant que ce sont des tâches relativement simples car je ne souhaite pas devoir courir en hâte à la poste le dernier jour.

Les objectifs pour le prochain jour sont donc les suivants :

- Etablir les schémas des class dont je connais déjà le besoin selon l'architecture que j'ai choisie pour la disposition de mes packages/class.
- Continuer la création du design de l'application. Il y a encore beaucoup à faire de ce côté-là.
- Essayer d'envoyer un mail depuis le serveur de production. Ça n'est pas planifié dans mes tâches pour le prochain jour mais le collaborateur qui a mis en place le serveur SMTP finit son stage à la caisse demain et je voudrais donc profiter de tester l'envoi de mail avant son départ, pour lui poser d'éventuelles questions sur la configuration de PostFix si je rencontre des problèmes.

Tâches réalisées

2 Réalisation
2.1 Base de données
2.1.1 Création — 100%
2.1.2 Test — 100%
2.2 Interface web
2.2.1 Création du design — 35%

Vendredi, 01 Mars 2013

Février
Lundi 25
Jeudi 28

Mars
Vendredi 01
Lundi 04
Jeudi 07
Vendredi 08
Lundi 11
Jeudi 14
Vendredi 15
Lundi 18
Jeudi 21
Vendredi 22
Lundi 25
Mardi 26
Mercredi 27

Périodes		Déroulement	
		Tâches effectuées	**Remarques sur les tâches effectuées**
07:30	08:30	Résolution du centrage des éléments, création d'une méthode pour récupérer le nom d'un interne sur la base de son adresse email.	J'ai utilisé le composant Embedded que j'ai inclus dans un VerticalLayout et défini l'alignement du composant.
08:30	09:30	Test d'envoi d'un mail depuis le poste de développement avec un serveur SMTP gratuit puis depuis le serveur.	J'ai tout d'abord rencontré un problème avec la connexion MySQL, j'utilisais dans mon code un paramètre qui n'existait pas, ce qui déclenchais une NullPointerException. Ensuite, j'ai réussi à envoyer des mails, mais seulement en local sur des adresses @avs.vs.ch… Il est important que je comprenne pourquoi ! Plus de détails ci-dessous.
09:30	10:30	Ecoute de l'URL avec un ParameterHandler et un UriFragmentUtility	Pour pouvoir détecter et récupérer les paramètres de l'URL. Le parameterHandler nous permet de récupérer les paramètres par ?parametres et l'UriFragmentUtility permet de détecter les ancres, sauf que j'ai un problème avec ce composant, il prend de la place et déforme mon design, je ne comprends pas pourquoi, je le désactive pour le moment puisque je n'en ai pas encore besoin.
10:30	11:30	Récupération et stockage des informations sur l'utilisateur et de l'URL	Création de plusieurs méthodes pour récupérer le nom du navigateur et sa version, le nom du système d'exploitation, l'adresse IP publique de l'utilisateur… et récupération de l'URL pour les envois par mail.
11:30	13:00	Envoi d'un mail crypté contenant de l'HTML et essai de récupération des informations de la base de données en fonction de la valeur cryptée	J'ai modifié la méthode d'envoi de mails pour lui permettre d'envoyer des e-mails en HTML. J'ai également créé une class pour le cryptage SHA256 que j'ai testée en envoyant par mail la valeur cryptée d'un mail. J'ai ensuite pu ressortir de la base de données les informations liées à l'adresse cryptée.
13:00	14:00	Pause de midi	
14:00	15:40	Design de l'application	J'ai voulu essayer de passer d'un design à l'autre (Design de connexion et design général) mais le problème est que le design de connexion se base sur un thème vaadin différent du design principal. Donc, quand on passe d'un design à l'autre, la page est rechargée et le ?restartApplication redémarre toute l'application, ce qui fait qu'on ne peut jamais afficher le design

			principal. Je suis obligé de ne pas redémarrer l'application pour pouvoir switcher d'un design à l'autre.
15:40	16:00	Apéro de départ du stagiaire EMVs	Le stagiaire de 3e année nous quitte et je participe à l'apéro quelques minutes tout en réfléchissant à une manière de switcher proprement entre les deux thèmes.
16:00	17:00	Design du contenu et des composants de l'application	Création du menu en accordéon sur la partie principale, création du module de gestion des contacts. Création d'une class SQLTable basée sur la class Table de Vaadin, ce composant sera destiné à recevoir les données sous forme de tableau, je lui ajoute un menu contextuel sur le clic droit.

Problèmes rencontrés & résolution

Envoi de mails depuis le serveur
Sur le serveur de production est installé un serveur SMTP PostFix. J'ai tout d'abord utilisé ce service en ligne de commande pour un premier test : tout va bien, mais il n'envoie qu'à des adresses locales, c'est-à-dire des adresses se terminant par @avs.vs.ch. J'ai ensuite testé l'envoi d'un mail dans mon code Java, depuis mon poste de développement, à l'aide d'un petit serveur SMTP gratuit (miniRelay). Là, je peux envoyer les mails en externe mais pas partout, je peux envoyer à dominique@roduit.com mais pas à dominikroduit@hotmail.com. En cherchant des informations sur internet, on trouve que plusieurs FAI bloquent les mails du port 25 s'ils détectent que l'IP utilisée pour envoyer l'email n'est pas dans le range autorisé par le FAI. Selon moi il y a donc deux problèmes ! Un qui vient de la DMZ qui ne laisse pas sortir les mails en externe et un autre qui vient du problème décrit ci-dessus. J'ai signalé ce problème au chef du groupe système-exploitation, il a envoyé un mail à l'état du Valais pour leur demander où en sont les configurations de la DMZ. Il me dit de ne pas rester bloquer sur ce problème, de continuer pour l'instant en utilisant seulement les adresses locales… C'est donc ce que je fais. La personne qui a configuré PostFix me dit que c'est normal que ça ne parte pas vers l'extérieur puisque la DMZ n'est pas encore configurée, ça me rassure mais je crains d'avoir d'autres problèmes une fois la DMZ configurée. Pour l'instant je continue en envoyant aux adresses locales.

Basculement d'un thème à l'autre
J'ai créé deux thèmes différents, l'un pour la connexion, l'autre pour le reste de l'application. J'utilise donc deux feuilles de styles différentes. Le problème est que lorsque je change de feuille de style, la page est rechargée et en gardant le ?restartApplication dans l'URL, toute l'application est réinitialisée et ainsi je n'arrive pas à basculer d'un thème à l'autre en conservant ce paramètre. Il faudra que je trouve un moyen. La solution de dernier secours sera de ne pas changer de feuille de style.

Objectifs pour le prochain jour

Comme je le pensais, je n'avais pas pris trop d'avance puisque je n'ai toujours pas terminé la partie Design, il reste encore toute la partie wizard/création des dossiers à mettre en place, les tableaux qui vont contenir les données, les formulaires d'ajouts/édition de contacts… Il y a encore beaucoup de travail. La partie navigation reste aussi un point qui risque de prendre du temps. Mon objectif pour le prochain jour est donc d'essayer de terminer au moins le design des wizard pour la création des dossiers.

Tâches réalisées

2 Réalisation
2.2 Interface web
2.2.1 Création du design 70%
2.2.2 Navigation 10%
2.3 Modules spécifiques
2.3.1 Identification utilisateur 20%
2.3.4 Envoi d'URL cryptées 40%

Lundi, 04 Mars 2013

Février
Lundi 25
Jeudi 28

Mars
Vendredi 01
Lundi 04
Jeudi 07
Vendredi 08
Lundi 11
Jeudi 14
Vendredi 15
Lundi 18
Jeudi 21
Vendredi 22
Lundi 25
Mardi 26
Mercredi 27

Périodes		Déroulement	
		Tâches effectuées	**Remarques sur les tâches effectuées**
07:30	08:30	**Design de l'application** Création de la page contacts avec un bouton d'ajout et une fenêtre pour l'ajout et l'édition.	Problème bloquant : Au premier chargement de la fenêtre, l'application s'affiche normalement. Dès le redémarrage de l'application, les requêtes sont exécutées mais la fenêtre flottante ne s'affiche pas. J'ai résolu le problème en allant chercher la fenêtre principale depuis sa propre class et non depuis la class contenant les objets en global.
08:30	09:30	**Design de l'application** Création de la fenêtre et de l'assistant 1, pour l'ajout d'un dossier.	Petit problème rencontré : je n'ai pas réussi du 1er coup à ajouter un Slider dans le FormFactory. On ne peut pas transtyper un composant de type Field en Slider, il faut retourner uniquement le Slider avant de passer dans le bloc conditionnel.
09:30	11:30	**Design de l'application** Création de l'assistant 2, pour l'ajout des destinataires dans un dossier	Un gros problème se présente à moi, je n'arrive pas à récupérer l'item sélectionné dans la ComboBox. Je perds une heure sur ce problème. J'ai cherché sur internet plusieurs manières de faire, sur le site de Vaadin également. Aucun moyen de récupérer la valeur sélectionnée dans le ComboBox. Je debug. J'arrive finalement à sélectionner l'objet Contact mais pas la valeur. Je demande de l'aide à Nicola Iannace. Après une dizaine de minutes nous arrivons à récupérer la valeur, il fallait transtyper la variable pour pouvoir récupérer ses propriétés.
11:30	13:00	**Design de l'application** Suite de la création de l'assistant 2, pour l'ajout des destinataires	Création d'un tableau tampon qui va réceptionner les éléments sélectionnés dans la ComboBox. Ajout d'un menu contextuel sur les éléments du tableau pour permettre la suppression d'items. Vérification : au minimum un contact doit être sélectionné. Création d'une méthode de récupération des entrées du tableau
13:00	14:00	Pause de midi	
14:00	15:00	**Design de l'application** Création de l'assistant 3, pour l'envoi des fichiers. Intégration de l'add-on EasyUploads.	Pour le téléchargement de fichiers, j'aurais pu utiliser directement le composant Vaadin « Upload » mais celui-ci n'est pas très adapté à mes besoins. Un add-on beaucoup plus proche de ce dont j'ai besoin existe. Il permet l'upload multiple de fichiers. Cependant, il y a un inconvénient, il ne fonctionne pas avec la version 7 de Vaadin.

			Heureusement, nous utilisons toujours la version 6 qui est plus stable.
15:00	16:30	Configuration de Tomcat pour l'upload de fichiers, configuration du composant d'upload multiple. Personnalisation du design du composant pour l'intégrer à l'application.	J'ai ajouté également une zone de drag&drop où l'utilisateur qui utilise un navigateur récent (Firefox, Chrome ou Safari) peut directement glisser les fichiers qu'il souhaite envoyer dans l'application.
16:00	17:00	La tâche « Création du design de l'interface web » est presque terminée, il ne manque plus qu'à construire l'aspect de l'intérieur d'un dossier, c'est donc ce que je commence à faire.	Je commence aussi à travailler sur le basculement d'une étape à l'autre de l'assistant de création d'un dossier.

Problèmes rencontrés & résolution

Au lancement de l'application tout se passait bien. Si je recharge l'application, même complètement en utilisant le paramètre URL ?restartApplication, je suis surpris de constater que lorsque je clique sur un des boutons de mon application, rien ne se passe, alors qu'une fenêtre flottante devrait s'afficher. Le problème venait de la récupération de la fenêtre principale depuis la class de conservation des objets en global.

Petit problème lors de la création de l'assistant pour l'ajout de destinataires dans un dossier : Pour ajouter un composant Slider, il faut le retourner avant de passer dans le bloc conditionnel de la fonction createField() qui renvoie un composant de type Field.

Problème persistant pendant une heure : je n'arrivais pas à récupérer la valeur sélectionnée dans la ComboBox. Avec l'aide de Nicola Iannace j'arrive finalement à la récupérer. Il suffisait de transtyper la valeur retournée en objet de type « Contact » pour pouvoir en récupérer les propriétés.

Objectifs pour le prochain jour

Cette journée m'a permis une progression conséquente. Je ne suis plus autant en avance sur la planification car la tâche « Création du design » m'a pris un temps énorme, je ne l'ai d'ailleurs pas entièrement terminée. Cependant je sais que le temps que j'ai pris pour réaliser cette partie me sera bénéfique pour la suite puisque je n'aurai plus qu'à mettre ensemble tout ce que j'ai construit et ajouter la partie dynamique (enregistrement et récupération des valeurs depuis la base de données, envoi des mails, automatisations de journalisation et d'archivage des dossiers).

Tâches réalisées

2 Réalisation
- **2.2 Interface web**
  - 2.2.1 Création du design — 95%
  - 2.2.2 Navigation — 14%
- **2.3 Modules spécifiques**
  - 2.3.3 Transfert de fichier — 10%

Jeudi, 07 Mars 2013

Février
Lundi 25
Jeudi 28

Mars
Vendredi 01
Lundi 04
Jeudi 07
Vendredi 08
Lundi 11
Jeudi 14
Vendredi 15
Lundi 18
Jeudi 21
Vendredi 22
Lundi 25
Mardi 26
Mercredi 27

Périodes		Déroulement	
		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	Design de l'application	Création de la page pour la visualisation des données d'un dossier. Je n'ai qu'ajouté le tableau qui répertorie les fichiers du dossier pour le moment.
08:30	09:30	Design de l'application	Ajout d'illustrations pour les extensions de fichiers dans le tableau qui répertorie les fichiers du dossier. Je ne les affiche pas pour le moment parce qu'une erreur indiquant que le parent de l'objet existe déjà est déclenchée et je préfère ne pas m'arrêter maintenant sur de petits détails. Création de la page d'accueil.
09:30	10:30	Séance du service informatique	
10:30	11:30	Design de l'application	Ajout d'un statut aux dossiers. Je fais maintenant la différence entre les dossiers archivés et ceux qui sont actuellement disponibles pour les destinataires.
11:30	13:00	Navigation à travers l'interface de l'application	Je commence la tâche de navigation sur l'interface. Je donne la possibilité de switcher d'un module à l'autre. J'ai encore quelques bugs au niveau de l'affichage des éléments qui ne se remplacent pas tout le temps ou qui se suppriment quand ils ne le devraient pas.
13:00	14:00	Pause de midi	
14:00	15:00	Validation du développement effectué auprès du responsable et avancement de la navigation	Nous avons fait le point pendant 15 minutes avec Mr. Lorétan, pour valider le développement effectué jusqu'ici. Il n'a pas de remarque particulière sur le design et le développement, il est très satisfait de l'avancement. J'ai également mis en place une méthode pour retraduire l'application. Elle fonctionne mais je dois recharger la page pour que les modifications prennent effet. Je n'ai pas encore mis au point le rechargement de la page pour l'instant puisque cette tâche ne fait pas partie intégrante du cahier des charges.
15:00	16:00	Corrections / Externalisation des textes et images	Enregistrement des textes de l'application dans la base de données. Regroupement des images dans la class Global en tant que variable accessible depuis n' importe où.
16:00	17:00	Test du design de l'application et de la navigation déjà réalisée	Rapport de test effectué pour tester les informations affichées/la partie de navigation déjà réalisée et corriger les erreurs en fonction des résultats obtenus.

Problèmes rencontrés & résolution

Lorsque j'ai voulu ajouter des icônes d'illustration des différents types de fichiers dans le tableau, j'ai vite été freiné par une erreur que je n'ai pas réussi à résoudre pour le moment. Cette erreur indique que le parent de l'objet existe déjà. Je ne sais pas encore ce que signifie ce message puisque je n'ai pas pris le temps de m'arrêter sur cette erreur, j'ai préféré continuer sur l'essentiel de mon cahier des charges car cela n'en faisait pas vraiment partie. J'y reviendrai à la fin si j'ai du temps.

Objectifs pour le prochain jour

Pour le prochain jour, la planification effectuée prévoit la poursuite de la tâche « Navigation ». La tâche de création du design est finalement terminée, elle m'a pris 1h de plus que prévu et m'a permis de me réajuster avec la planification initiale pour la prochaine tâche. Cependant, plusieurs tâches ont déjà été entamées et bien qu'elles ne soient pas entièrement fonctionnelles, elles me permettront de garder l'avance que j'avais prise dès le départ.

Tâches réalisées

2 Réalisation
2.2 Interface web
2.2.1 Création du design — 100%
2.2.2 Navigation — 45%
2.2.3 Test — 35%
2.3 Modules spécifiques
2.3.1 Identification utilisateur — 20%

Vendredi, 08 Mars 2013

Février
Lundi 25
Jeudi 28

Mars
Vendredi 01
Lundi 04
Jeudi 07
Vendredi 08
Lundi 11
Jeudi 14
Vendredi 15
Lundi 18
Jeudi 21
Vendredi 22
Lundi 25
Mardi 26
Mercredi 27

Périodes		Déroulement	
		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:40	Navigation Transition des layout Connexion/Interface	Pour permettre une navigation plus rapide, plus fluide et sans rechargement de la page lors de la transition des designs Connexion/Interface utilisateur, j'ai dû copier les images et styles du thème Reindeer, mais uniquement pour les composants dont j'avais besoin, c'est-à-dire la fenêtre de type Reindeer.Black et tous les composants qui y sont intégrés.
08:40	10:00	Navigation Première connexion d'un utilisateur, données manquantes, cas d'erreurs	Lors de la première connexion, l'utilisateur n'a aucune donnée, il faut donc adapter l'interface à cette situation. J'ai donc ajouté quelques images pour « conduire » l'utilisateur : j'affiche visuellement quelles sont les étapes à effectuer pour pouvoir transférer des fichiers et une image pour indiquer si l'étape est complète ou non. Si l'utilisateur n'a aucun contact et qu'il appuie sur « Nouveau dossier », on lui dit qu'il doit tout d'abord ajouter des contacts et on le redirige directement vers la page de création des contacts.
10:00	11:30	Navigation Suppression des messages de test Rédaction et mise en forme des pages « Conditions d'utilisation » et « A propos »	Depuis les pages « Conditions d'utilisation » et « À propos », il n'était pas possible de revenir à la page précédente, il fallait cliquer sur l'onglet du menu qui n'était pas encore sélectionné. C'est pourquoi j'ai ajouté un petit bouton qui permet le retour en arrière dans la navigation. Ce bouton va contrôler quel onglet est sélectionné et va charger le contenu correspondant. Plusieurs messages que j'ai insérés pour les tests me semblent inutiles pour les futurs utilisateurs de l'application, je les supprime donc et mets en forme les messages que je conserve (notification ou erreur)
10:30	12:00	Navigation Disponibilité d'un dossier, taille total des fichiers du dossier	Récupération de la disponibilité d'un dossier par la différence entre sa date d'expiration et de création. Récupération du nombre de fichiers d'un dossier et de la taille totale des fichiers contenus dans ce dossier.
12:00	13:00	Préparation des tests à réaliser	J'ai rédigé un document ou je note chaque fois ce que je dois tester, de cette manière je n'aurai plus qu'à effectuer les tests planifiés pour vérifier que tout fonctionne.
13:00	14:00	Pause de midi	
14:00	15:00	Navigation / Envoi d'URL cryptée Envoi du mail pour le téléchargement d'un dossier	Enregistrement des paramètres utiles dans la base de données et envoi du mail avec un paramètre contenant pour l'instant uniquement une valeur indiquant à la page que l'on est en mode téléchargement de fichiers et qu'il faut charger une page spécifique. J'ai eu des problèmes de pointeur null au départ mais je les ai résolus en re-factorisant la

			méthode de création du layout principal. Pour l'accès normal à l'interface, je charge le layout et les composants. Pour le téléchargement, je ne charge que le layout et je rajoute par-dessus les éléments spécifiques à la page de téléchargement.
15:00	16:00	Navigation	Adaptation de la page « À propos ». Son bouton « Retour » doit détecter si nous sommes sur une page de téléchargement ou non, car il ne recharge pas les mêmes composants dans les deux cas.
16:00	17:30	Création des composants de la page de téléchargement	La page de téléchargement utilise le même design que le reste de l'application mais c'est une page à part qui doit être accessible à n'importe qui. Même à des utilisateurs non-inscrits. Cette page doit présenter les informations sur les fichiers, permettre leur téléchargement, et journaliser les actions des utilisateurs. En réalisant cette page, je me dis qu'il faudrait que je rajoute une table dans ma base de données pour journaliser aussi la simple consultation du dossier de téléchargement par un des destinataires. Cette table nous permettrait ensuite de pouvoir notifier à l'auteur du transfert que le dossier a été visité mais que les fichiers n'ont pas été téléchargés. Cette une idée d'amélioration, elle ne fait pas partie du cahier des charges.

Problèmes rencontrés & résolution

Aucun problème ne m'a fait perdre beaucoup de temps aujourd'hui. J'ai eu quelques difficultés pour afficher la page de téléchargement des fichiers mais j'y suis finalement arrivé. Je n'ai toujours pas résolu le problème indiquant que « le parent de l'objet existe déjà ».

Objectifs pour le prochain jour

Terminer la navigation, résoudre les problèmes qui pourraient survenir lorsque je testerai l'interface après avoir ajouté la page de téléchargement.

Donner la possibilité à l'utilisateur de pouvoir télécharger un fichier, je n'ai pas encore réussi à le faire à cause du problème de composant cité ci-dessus. J'aurai certainement besoin de l'aide de Nicola pour ce point.

Effectuer tous les tests décrits dans mon rapport de test et finir sa rédaction.

Terminer l'identification utilisateur et journaliser tous les cas d'erreur.

Tâches réalisées

2 Réalisation
- **2.2 Interface web**
  - 2.2.1 Navigation — 80%
  - 2.2.2 Test — 50%
- **2.3 Modules spécifiques**
  - 2.3.1 Identification utilisateur — 60%
  - 2.3.4 Envoi d'URL cryptées — 50%

Lundi, 11 Mars 2013

Février / Mars

Lundi 25 — Jeudi 28 — Vendredi 01 — Lundi 04 — Jeudi 07 — Vendredi 08 — **Lundi 11** — Jeudi 14 — Vendredi 15 — Lundi 18 — Jeudi 21 — Vendredi 22 — Lundi 25 — Mardi 26 — Mercredi 27

Périodes		Déroulement	
		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	Mise à jour de la JavaDoc et finalisation du module de connexion	Premier essai de génération de la JavaDoc pour voir ce que ça donne en l'état actuel. Mise à jour de la JavaDoc et suppression des tags HTML bloquant la génération (des balises que j'avais oublié de refermer dans une classe).
08:30	09:30	Ajout du bouton « Télécharger » en face de chaque fichier, dans la page de téléchargement	J'ai enfin réussi à résoudre le problème qui m'indiquait que l'objet avait déjà un parent et j'ai donc pu ajouter à mon tableau le bouton de téléchargement des fichiers. Il y a eu un autre problème qui s'est posé : Vaadin ne dispose d'aucun composant avec l'aspect visuel du bouton, permettant de lancer une URL. J'ai donc créé un composant ButtonLink qui permet de le faire.
09:30	10:30	Test sur le serveur pour trouver le dossier d'upload	J'ai exporté mon application sur le serveur pour pouvoir tester l'upload et pouvoir repérer le dossier qui contient les fichiers uploadés par défaut. Ce dossier est en fait la racine de la JRE. Ce n'est donc pas là que nous voulons stocker nos fichiers, je cherche un emplacement ou mes fichiers pourront être accessibles depuis l'application puisque les utilisateurs doivent pouvoir les télécharger par un simple lien. J'opte finalement pour le répertoire temporaire d'upload par défaut de tomcat, je modifie uniquement le chemin de destination pour que mes fichiers soient accessible depuis le net. Plus d'informations dans la documentation technique.
10:30	11:15	Visite intermédiaire des experts	J'ai pu montrer aux deux experts le travail effectué jusqu'ici, je leur ai expliqué brièvement comment j'avais structuré mes class, comment était construite la base de données,… Je leur ai fait une démonstration en partant d'un utilisateur non inscrit en allant jusqu'au partage d'un dossier. Je leur ai demandé une nouvelle fois si tout le code était nécessaire en annexe mais ils me confirment que oui. Je demande également s'il est possible d'ajouter des tâches au cahier des charges mais ils me suggèrent de terminer avant tout toutes celles qui ont été prévues initialement et de bien insister sur la documentation, de rajouter des petits plus à la fin si vraiment je suis en avance.
11:15	13:00	Ajout d'icônes d'illustration aux tableaux contenant les fichiers. Création de méthodes pour la récupération de liens de téléchargement.	J'ai réussi à résoudre le problème que j'avais pour ajouter une colonne contenant des composants. Il suffisait d'utiliser la méthode addGeneratedColumn du composant Table. J'ai eu quelques problèmes pour récupérer l'URL du fichier sur le serveur. J'ai décrit dans ma documentation technique comment je l'avais fait mais j'ai dû modifier la racine de contexte du dossier temporaire de tomcat. Les méthodes utiles à la récupération du dossier d'upload sur le serveur sont getUploadDir et getURLForFile. Elles sont localisées dans Utilities.
13:00	14:00	Pause de midi	
14:00	15:30	Modification de la navigation pour le wizard de création d'un dossier	Initialement, j'avais fait qu'avant même de passer au wizard suivant, on créait les éléments nécessaires, on insérait par exemple dans la base de données le dossier dès le premier wizard, on créait les destinataires au deuxième, … Cela posait un réel problème : si l'utilisateur rechargeait la page et qu'il n'avait pas ajouté de fichier ou

			aucun destinataire, on se retrouvait avec des dossiers vide… Maintenant, tout est créé lorsqu'on appuie sur le bouton « Terminer » dans le dernier wizard. Ça évitera beaucoup d'erreurs et l'utilisateur peut annuler l'opération à n'importe quel moment.
15:30	16:30	Création d'une archive	Création d'une archive au format ZIP à la fin de la création d'un dossier. Je n'arrive pas encore à le faire. Le fichier ZIP est correctement créé mais un problème se pose : les noms de fichiers. Je ne récupère pas le vrai nom du fichier tel qu'il est enregistré sur le disque mais le nom du fichier que l'utilisateur a donné sur son PC. Ce n'est pas la bonne méthode je dois corriger cela !
16:30	17:30	Obtention des noms de fichiers tels qu'ils sont enregistrés + re-nommage	Création de deux méthodes dans la class Utilities. Une pour supprimer tous les accentuations d'une chaine de caractères et une deuxième qui utilise cette dernière pour formater le nom de fichier. Les fichiers sont maintenant rendu uniques : en fin de nom, ils contiennent la date au format yyyyMMddHHmmss pour éviter les doublons. Ce format sera utilisé pour le téléchargement des fichiers, le nom d'affichage sera le nom de fichier que l'utilisateur aura donné à son fichier.

Problèmes rencontrés & résolution

J'ai rencontré passablement de problèmes, presque tous étaient liés et concernaient l'upload de fichier. Comme ce n'étaient pas les bons noms qui étaient enregistrés, ce n'étaient pas non plus les bons fichiers qui étaient téléchargés. On arrivait donc à des cas d'erreur ou le fichier n'existait pas ou il n'était pas accessible parce que l'accentuation des fichiers Windows n'était pas prise en compte sur le serveur Linux. Ceci est donc en voie de correction.

Objectifs pour le prochain jour

J'ai atteint un des jalons de ma planification aujourd'hui. Je me lance dans la dernière partie de la réalisation qui est la mise en place de tous les modules spécifiques. Presque tous ont déjà été entamés, les gros points qui doivent encore être travaillés sont la journalisation des opérations, le transfert de fichiers qui est quand même le principal objectif de ce projet, le téléchargement des fichiers et bien sûr, la documentation.

A ce jour, je continue de ne tester l'application qu'avec des adresses e-mail internes car la DMZ n'est apparemment toujours pas fonctionnelle. Je ne peux malheureusement rien y faire, ce que nous avons pu faire dans l'entreprise nous l'avons fait consciencieusement avant le début du TPI, c'est maintenant à l'état du Valais de faire sa part du travail mais je n'ai toujours pas de nouvelles les concernant.

Tâches réalisées

2 Réalisation
- **2.2 Interface web**
  - 2.2.1 Navigation — 100%
  - 2.2.2 Test — 100%
- **2.3 Modules spécifiques**
  - 2.3.1 Identification utilisateur — 100%
  - 2.3.3 Transfert de fichiers — 20%
  - 2.3.4 Envoi d'URL cryptées — 60%
  - 2.3.5 Téléchargement des fichiers — 30%

Jeudi, 14 Mars 2013

Février
Lundi 25
Jeudi 28

Mars
Vendredi 01
Lundi 04
Jeudi 07
Vendredi 08
Lundi 11
Jeudi 14
Vendredi 15
Lundi 18
Jeudi 21
Vendredi 22
Lundi 25
Mardi 26
Mercredi 27

Périodes		Déroulement	
		Tâches effectuées	Remarques sur les tâches effectuées
07:30	09:00	Essai d'envoi d'un mail en externe Récupération du nom original du fichier lors d'un transfert	J'ai reçu un mail de Mr. Arcudi ce matin, une copie du mail du responsable système de l'état du valais lui disant que l'adresse https://cccvstransfert.vs.ch/ était atteignable, mais seulement depuis l'extérieur comme convenu. Je n'ai pas donc pas pu le tester directement mais nous avons réussi à atteindre la page d'accueil de Tomcat en testant le lien depuis le réseau d'un mobile. Les problèmes qui se posent maintenant sont les suivant : comment atteindre mon application, sur quel port ? (car nous n'avons pas réussi à l'atteindre depuis l'extérieur, nous avons eu un message « délai d'attente de la demande dépassé »), comment afficher mon application directement, sans passer par la page tomcat ? Comment tester l'envoi de mail à des adresses externes sans avoir accès depuis le réseau de la caisse de compensation ? Je contacte Monsieur Frédéric Follonier, responsable système à l'état du Valais, pour lui poser ces questions.
09:00	10:30	Modification du script d'upload Création de l'archive des fichiers	A la fin de l'upload, on enregistre maintenant les noms originaux des fichiers + leur nom d'enregistrement sur le disque. Cela nous permet de pouvoir créer des archives sans problèmes liés au nom des fichiers et d'éviter également les problèmes d'accentuation, d'espaces, de majuscules… J'ai également dû modifier la façon de récupérer les fichiers puisque si l'utilisateur supprime des fichiers du tableau il faut bien entendu que les modifications soient répercutées dans l'ArrayList qui les contient tous les fichiers chargés. On peut maintenant créer sans problème une archives ZIP des fichiers envoyés et les noms très longs sont formatés.
10:30	12:00	Page de téléchargement Traduction de l'application	Modification du module de traduction. Correction : Lorsque le dossier a expiré, l'accès n'est plus possible. Modification de l'affichage des informations du dossier (intégration dans le menu de gauche). Application du lien sur le bouton pour le téléchargement de l'archive du dossier. Affichage d'une zone pour l'affichage de la journalisation des actions.
12:30	13:00	Sauvegarde des actions (journalisation) Affichage des journalisations au chargement Modification de la base de données	Les actions de téléchargement sont maintenant journalisées correctement, il reste encore à journaliser les actions de consultation des dossiers. On distingue le téléchargement d'un fichier ou de l'archive d'un dossier grâce à la clé étrangère du fichier dans la table. Si la clé étrangère est nulle, c'est une archive qui a été téléchargée. J'ai dû modifier la table de journalisation des téléchargements pour pouvoir également enregistrer et distinguer les actions de consultation. J'ai donc renommé la table en question et ajouté un champ pour faire la

			distinction entre les différents types d'actions. Au chargement de la page, le tableau des actions est rempli par les actions enregistrées et les nouvelles actions sont ajoutées à la suite.
13:00	14:00		Pause de midi
14:00	15:00	Journalisation des consultations Réflexion pour la suite : je teste l'exécution d'un fichier java depuis un fichier Shell	Journalisation des consultations et affichage de la date dans le tableau des actions Pour anticiper la suppression des fichiers et m'éviter des surprises (comme je n'ai encore jamais utilisé la Crontab pour exécuter des scripts java), je teste l'exécution d'un script java depuis un fichier Shell, puisque ce sera un fichier Shell appelé depuis la Crontab. Le résultat a l'air plutôt concluant mais je n'arrive pas à compiler directement mes classes java depuis le serveur, je concevrai donc les fichiers en local et les exporterai sur le serveur pour les tests.
15:00	16:30	Tests sur le serveur	Là, un gros problème survient, je n'arrive pas à télécharger un seul fichier, aucune archive n'est créée… ça à l'air catastrophique au premier abord et l'erreur prend du temps à être résolue, je trouve finalement que les fichiers ne sont pas renommés sur le disque. Ils sont donc enregistrés avec le nouveau nom mais ce nom n'existe pas physiquement, c'est parce que je renseignais un faux chemin pour renommer mon fichier ! Lorsque je veux atteindre et manipuler un fichier sur Linux, je dois donner les chemins absolus ! Le problème est résolu pour les fichiers, mais il me reste encore à résoudre la création de l'archive, le problème doit surement être similaire.
16:30	17:30	Correction des problèmes de chemins pour la sauvegarde et le téléchargement des fichiers	J'ai finalement repassé en revue toutes mes variables et méthodes qui me retournaient des chemins de destination pour le téléchargement ou le dépôt de fichier. J'ai créé une variable global qui me permet de savoir si je suis en développement ou en production, et je définis les variables de chemin en fonction de ce paramètre. J'arrive maintenant à créer l'archive, les fichiers et télécharger le tout, sur le poste de développement, et sur le serveur de production.
20:30	21:00	Test de l'application en ligne depuis la maison	L'application est en disponible à l'adresse https://cccvstransfert.vs.ch/. Ce n'est pas la version finale définitive mais comme elle n'est accessible que depuis le réseau externe, je profite de la tester depuis la maison, j'envoie une dizaine de fichier, je teste la rapidité de l'upload avec des fichiers volumineux, c'est encore relativement performant, en 5 minute j'envoie environ 160 Mo.

Problèmes rencontrés & résolution

Téléchargement des fichiers
Un gros problème s'oppose à moi aujourd'hui par rapport au téléchargement des fichiers. Je n'ai plus testé l'application sur le serveur depuis une journée et de grosses modifications ont été effectuées. En local tout marche bien mais lorsque j'exporte mon projet vers le serveur, je suis surpris de remarquer que celui-ci n'enregistre pas les fichiers au bon endroit, il ne les renomme pas, il ne récupère pas les bons noms de fichiers… Tout cela rend le téléchargement des fichiers impossible. Il m'a fallu beaucoup de temps pour dépanner ce problème de chemin de destination, j'utilisais une méthode qui ne me retournait pas le bon chemin de destination vers les fichiers envoyés sur le serveur. J'ai finalement pu résoudre le problème simplement, en créant une variable global qui m'indique si je suis en production ou en développement. Grâce à celle-ci, j'ai pu simplifier mes méthodes et avoir une vision beaucoup plus claire de mon problème, j'ai ensuite pu le résoudre efficacement.

Envoi des messages électroniques vers des adresses externes
Toujours le même problème d'envoi des mails vers des adresses extérieures persiste malgré la mise en fonction de la DMZ. J'ai notifié le responsable du groupe système de l'état du valais de mon problème, je lui ai exposé la situation et lui ai demandé si ce problème était normal, s'il était lié à eux, s'il était en rapport avec la DMZ ou non. J'attends maintenant sa réponse mais à mon avis le problème vient de PostFix qui n'est pas configuré pour laisser sortir les mails sur internet, dans ce cas je ne sais pas ce que je dois faire, prioriser la résolution de ce problème qui ne fais pas partie intégrante des tâches de mon cahier des charges ? Je préfère avant tout terminer toutes les tâches que je dois réaliser avant de m'attarder sur ce problème qui d'après moi ne devrait pas exister puisqu'il avait été convenu que tout soit effectif avant le début du projet.

Objectifs pour le prochain jour

Commencer le script de suppression automatique des fichiers, laisser la possibilité à l'utilisateur de pouvoir modifier les noms de ses fichiers et pouvoir leur donner une description. Eventuellement déjà afficher à l'utilisateur quel sont les fichiers qui sont téléchargés par les contacts agréés. Si j'arrive à faire tout cela, je m'attarderai également sur l'upload de fichiers. Je dois terminer les méthodes pour le blocage de la taille maximum d'un transfert et ce ne sera pas évident étant donné que nativement le transfert démarre automatiquement sans qu'une méthode puisse le stopper.

Tâches réalisées

2 Réalisation
2.3 Modules spécifiques

2.3.2 Journalisation des opérations	70%
2.3.3 Transfert de fichier	50%
2.3.5 Téléchargement des fichiers	80%

Vendredi, 15 Mars 2013

Février
Lundi 25
Jeudi 28

Mars
Vendredi 01
Lundi 04
Jeudi 07
Vendredi 08
Lundi 11
Jeudi 14
Vendredi 15
Lundi 18
Jeudi 21
Vendredi 22
Lundi 25
Mardi 26
Mercredi 27

Périodes		Déroulement	
		Tâches effectuées	**Remarques sur les tâches effectuées**
07:30	08:30	Création du script de suppression automatique des fichiers	Création d'un nouveau projet Java pour la suppression automatique des fichiers et des archives des dossiers expirés. Pourquoi un nouveau projet ? Car c'est une application à part qui devra être appelée depuis la crontab, elle ne sera pas accessible sur internet, elle n'est exécuté qu'en local.
08:30	10:30	Compilation du script de suppression en jar Tests sur le serveur Recherche des problèmes	Il a été très simple au premier abord de créer un fichier JAR, mais les complications sont arrivées lorsque j'ai dû inclure les librairies pour accéder à mysql depuis mon code. Bien que j'aie indiqué les ressources correctement dans la classpath, j'avais toujours une erreur affichée : ClassNotFoundException : com.mysql.jdbc.Driver. J'ai sué sur cette erreur que je ne comprenais pas étant donné que tout était configuré correctement. J'ai essayé d'inclure mes librairie dans le dossier lib du dossier racine de java, j'ai essayé de renseigner les ressources du classpath dans les options de la commande jar, rien n'a marché. Le problème a été résolu en exportant un JAR EXECUTABLE et non un JAR simple. L'archive JAR peut être exécutée et elle fonctionne bien, il reste maintenant à créer un script qui lance ce jar et à planifier une tâche automatique dans la crontab.
10:30	11:30	Correction : Date d'expiration et durée de vie restante d'un dossier	Toutes les dates de l'application n'étaient pas affichées correctement ! Si le dossier expirait en mars, c'était indiqué qu'il allait expirer en avril, la correction était simple, il suffisait de prendre l'élément précédent du tableau contenant les mois en textes. Avant cela, c'était la disponibilité du dossier qui était affichée – Le nombre de jour durant lesquels l'utilisateur avait choisi de mettre à disposition son dossier – maintenant, c'est le nombre de jour restant qui est indiqué, et s'il ne reste que quelques heures au dossier, le nombre d'heures restantes avant archivage.
11:30	13:00	Création du script shell pour l'appel du JAR de suppression des fichiers. Planification de l'appel du script dans la crontab et test. Ajout d'un champ « description » pour les dossiers et modification du mail d'envoi du	J'ai planifié dans la crontab l'exécution de mon script, il est correctement exécuté, un log de l'exécution se place dans /var/mail/root automatiquement. J'ai ajouté un champ « Description » ou l'utilisateur qui créé un dossier peut entrer une description du dossier en plus du nom qu'il lui donne dans la première fenêtre du wizard. Le mail qui envoie les liens de téléchargement a également été modifié en conséquence.

		lien de téléchargement	
13:00	14:00	Pause de midi	
14:00	15:30	Ajout d'un wizard pour le re-nommage des fichiers envoyés	A la fin de l'envoi des fichiers, nous avons maintenant la possibilité d'afficher un dernier wizard facultatif pour pouvoir modifier le nom des fichiers et leur attribuer une description. La sauvegarde des modifications n'est pas encore implémentée
15:30	16:30	Discussion à propos de l'envoi de mail avec Philippe Récupération des noms de fichiers renommés	J'ai reçu une réponse de Monsieur Frédérique Follonier mais je ne comprends pas vraiment ce que je dois faire alors je discute avec Philippe Arcudi pour savoir s'il comprend ce que je dois faire, il me dit de téléphoner à Monsieur Follonier directement car il n'a jamais fait de relai SMTP. Je le ferai lundi matin. Je tente de récupérer les noms des fichiers renommés mais je n'ai pas encore réussi à le faire.
16:30	17:00	Obtention des noms des fichiers uploadés : nom original, nom sur le disque, nom donné par l'utilisateur + descriptions	Il a fallu que je repense ma class pour le stockage des fichiers uploadés. J'ai rajouté des champs qui me permettent de récupérer beaucoup plus simplement les fichiers pour que la liste soit mise à jour lorsqu'un fichier est supprimé, et que les suppressions soient également prises en compte lorsqu'on passe en mode « Renommage ».

Problèmes rencontrés & résolution

Exportation du projet JAR pour la suppression automatique des fichiers

Bien que j'aie indiqué les ressources correctement dans la classpath, j'avais toujours une erreur affichée : ClassNotFoundException : com.mysql.jdbc.Driver. Cette erreur que je ne comprenais pas m'a fait perdre passablement de temps. Tout était pourtant correctement configuré. J'ai essayé d'inclure mes librairies dans le dossier lib à la racine du répertoire java, de renseigner les ressources du classpath dans les options de la commande jar, rien n'a fonctionné. Le problème a été résolu en exportant depuis Eclipse un JAR EXECUTABLE et non un JAR simple.

Objectifs pour le prochain jour

Je perçois au loin la note ultime de ce projet, je m'en approche à vive allure. Le projet est en bonne voie, il est quasi-entièrement fonctionnel, il ne reste qu'à résoudre quelques légers détails pour finaliser intégralement le cahier des charges. Je demeure néanmoins circonspect car d'infimes difficultés peuvent rapidement se transformer en de gros tracas qu'il convient de savoir gérer en relativisant.

Mes objectifs pour le jour prochain seront de limiter les sessions d'upload à 1 Go, c'est une partie que je trouve compliquée, je n'ai pas encore trouvé de moyen d'interrompre le transfert avant qu'il ne commence, le démarrage du transfert est ordonné automatiquement par les class de la librairie EasyUploads.

Tâches réalisées

2 Réalisation
2.3 Modules spécifiques
2.3.3 Transfert de fichier — 60%
2.3.5 Téléchargement des fichiers — 100%
2.3.6 Suppression automatique des fichiers — 45%

Lundi, 18 Mars 2013

Février
Lundi 25
Jeudi 28

Mars
Vendredi 01
Lundi 04
Jeudi 07
Vendredi 08
Lundi 11
Jeudi 14
Vendredi 15
Lundi 18
Jeudi 21
Vendredi 22
Lundi 25
Mardi 26
Mercredi 27

Périodes		Déroulement	
		Tâches effectuées	Remarques sur les tâches effectuées
08:00	09:30	Envoi de mails pour le règlement de plusieurs points administratifs concernant le projet. Paramétrage de la crontab pour le lancement du script de suppression chaque heures.	J'ai tout d'abord tenté de téléphoner à Monsieur Follonier pour lui poser des questions à propos de la configuration PostFix pour le relai SMTP mais celui-ci n'était pas disponible, je lui ai donc envoyé un mail pour lui demander de l'aide, j'ai également envoyé un mail au helpdesk de l'état du Valais pour leur demander une autorisation de sortie (comme m'a dit de faire Mr. Follonier dans son dernier mail). J'ai aussi corrigé un peu ma documentation technique après l'avoir soumise à tous mes collègues du service informatique.
09:30	10:30	Rédaction de la documentation technique	Jusqu'ici j'ai surtout insisté sur le code. Aujourd'hui j'ai pris un peu plus de temps pour bien ajouter dans ma documentation certains points importants, j'en ai profité pour mettre à jour les sections que je n'avais pas encore terminées.
10:30	11:30	Suppression automatique des fichiers	J'ai créé une méthode pour préparer l'envoi de mails à l'auteur et aux destinataires d'un dossier de transfert, lorsque celui-ci arrive au terme de sa durée de vie. Je n'ai pas encore écrit la requête qui sélectionnera les dossiers en question car je préfère terminer avant tout l'envoi de mail lors de la suppression des fichiers d'un dossier expiré. C'est une tâche qui est plus importante car elle est mentionnée dans mon cahier des charges.
11:30	13:00	Mise en forme des mails notifiant de la suppression et récupération des actions journalisées pour l'envoi d'un historique par mail.	Lorsqu'un dossier expire, les fichiers sont non seulement supprimés mais l'auteur reçoit un mail qui doit lui donner un « historique » des actions effectuées par les contacts sur le dossier. Je n'ai pas encore implémenté cet historique, par contre les contacts, eux, reçoivent déjà leur historique personnel des actions qu'ils ont effectuées sur le dossier.
13:00	14:00		Pause de midi
14:00	15:30	Envoi des mails de rappel pour les dossiers qui vont expirer dans moins de 24h.	Un mail est maintenant envoyé pour rappeler aux utilisateurs que le dossier sera supprimé. Il contient également un historique résumé et détaillé comme pour l'avertissement de suppression. Exportation du projet JAR pour la suppression et test sur le serveur. Petite erreur déclenchée car je n'avais pas mis à jour la base de données sur le serveur et certains paramètres que j'utilisais n'existaient pas.

15:30	16:30	Ajout d'un bouton et d'une fenêtre pour l'affichage du manuel utilisateur Documentation des class d'upload de fichier	Dans la page « Conditions d'utilisations », un bouton a été ajouté à droite, il permet de visionner le « Manuel d'utilisation » de l'application au format PDF, dans une fenêtre modal. Rédaction de la JavaDoc pour les classes du package common.component.upload (classes pour le transfert de fichier)
16:30	17:00	Limitation de la taille de la queue de transfert	J'ai commencé à mettre en œuvre la limitation de la taille maximum des fichiers sélectionnés par les utilisateurs. Je retourne dès la sélection des fichiers un code d'erreur 2 si les fichiers sélectionnés sont trop volumineux. Par contre j'ai une erreur qui survient si je choisis un seul fichier plus gros que 2 Go car l'add-on « EasyUploads » récupère la taille des fichiers dans une variable de type « int », qui ne peux que contenir des valeurs dans un intervalle de -2'147'483'648 à 2'147'483'647 alors que 2 Go = 2'147'483'648 octets, il faudra donc trouver le moyen de recompiler l'add-on en utilisant une variable de type « long », mais ensuite nous auront un problème de lenteur lors de l'envoi car les valeurs seront stockées sur 64 bits au lieu de 32, je verrai si je trouve un autre moyen…

Problèmes rencontrés & résolution

Limitation de la taille des fichiers envoyés
Si on choisit un seul fichier > 2 Go, l'application plante car la taille du fichier, stockée dans une variable de type « int » dépasse la valeur maximale que peut contenir ce type de variable ($\frac{2^{32}}{2}$). Il faudrait trouver le moyen de vérifier la taille en utilisant un type « long » avant l'envoi et de refuser le fichier si celui-ci fait plus de 2 Go, mais le problème est que cette manipulation est faite à l'intérieur des classes de l'add-on EasyUploads, que je ne peux pas retoucher à ma convenance. Où je m'expose à de gros risques que l'add-on ne fonctionne plus correctement… Je n'ai pas encore de solution pour ce problème actuellement.

Objectifs pour le prochain jour

Joindre Mr. Follonier pour résoudre ce problème d'envoi de mail car celui-ci commence à devenir urgent. Ce serait bien dommage que l'application n'envoie pas les mails aux adresses extérieures, bien que pour rappel, le fait que le serveur SMTP fonctionne n'était pas compris dans mon cahier des charges, ceci devait être effectif avant le début de la réalisation et c'est le stagiaire de 3e année qui avait été chargé de cette tâche.
Terminer la limitation de l'upload à 1 Go et trouver une solution au problème expliqué ci-dessus.
Me documenter sur la génération de certificat SSL pour la navigation sécurisée.

Tâches réalisées

2 Réalisation
2.3 Modules spécifiques
2.3.3 Transfert de fichier — 75%
2.3.6 Suppression automatique des fichiers — 85%

Jeudi, 21 Mars 2013

Février
Lundi 25
Jeudi 28

Mars
Vendredi 01
Lundi 04
Jeudi 07
Vendredi 08
Lundi 11
Jeudi 14
Vendredi 15
Lundi 18
Jeudi 21
Vendredi 22
Lundi 25
Mardi 26
Mercredi 27

Périodes		Déroulement	
		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	Récupération des fichiers	Lorsque je suis arrivé ce matin, le disque sur lequel je conserve mes fichiers n'était plus détecté et je n'arrivais plus à y accéder. Heureusement, j'ai plusieurs sauvegardes et un ordinateur de secours, mais j'ai quand même perdu pas mal de temps à pouvoir récupérer et synchroniser mon projet sur le poste de développement. Hier soir j'ai perdu une journée complète de documentation, le fichier ne s'ouvrait plus, il était corrompu…
08:30	10:30	Documentation et recherches pour la création d'un certificat SSL	J'ai réussi à créer un certificat SSL mais je n'ai pas encore réussi à l'intégrer à Tomcat. Je l'ai fait sur le poste de développement avant tout, car la méthode pour le faire sur le serveur est normalement exactement la même.
10:30	11:30	Limitation des extensions de fichiers lors de l'upload	Je repars sur ce que j'ai planifié pour aujourd'hui, c'est-à-dire, la limitation des extensions de fichiers. J'arrive maintenant à le faire correctement, sauf que j'ai trouvé un autre bug : lorsque j'envoie les fichiers en plusieurs sélections et que je veux supprimer un fichier du tableau, le fichier n'est pas supprimé sur le disque car ce n'est pas le bon nom de fichier qui m'est retourné. Je commence à résoudre cette erreur.
11:30	13:00	Limitation de la taille de l'envoi des fichiers	Visiblement, j'avais encore beaucoup de problèmes avec la limite de taille d'envoi des fichiers. Des erreurs survenaient lorsqu'on sélectionnait un fichier pendant qu'un autre était déjà en chargement. Lorsqu'on choisissait des fichiers trop volumineux et qu'ensuite on sélectionnait de petits fichiers, les petits fichiers obtenaient un index qui n'existait pas dans le tableau… Il y avait plein de petits problèmes comme cela que j'ai réglés. J'ai effectué les modifications uniquement pour la sélection de fichiers via le bouton « Parcourir … », je dois encore les faire pour la zone de glisser-déposer.
13:00	14:00	Pause de midi	
14:00	15:30	Limitation de la taille de l'envoi des fichiers	Affichage d'un indicateur qui montre à l'utilisateur la taille totale des fichiers envoyés et la taille totale autorisée. Lorsque la taille des fichiers dépasse 1 Go, l'utilisateur passe directement en mode « Renommage » et n'a plus la possibilité d'envoyer d'autres fichiers. Mise à jour de la taille des fichiers du tableau lors de la suppression d'un fichier. Lorsque tous les fichiers du tableau sont supprimés, je cache le bouton « Renommage ».
15:30	17:00	Ajout d'un indicateur de progression global	J'ai ajouté un indicateur de progression globale au fond de la fenêtre d'upload. Celui-ci permet de visualiser la progression de l'ensemble des fichiers en cours d'envoi. J'ai également amélioré l'aspect des barres de progression des fichiers en cours d'envoi dans le tableau.

Problèmes rencontrés & résolution

Concernant l'envoi d'e-mails en externe
Je me suis adressé à Philippe, chef du service informatique qui est également le responsable du groupe exploitation-système à propos du problème d'envoi des mails. Je lui ai demandé s'il pouvait s'en occuper à ma place car cette tâche ne fait pas partie de mon cahier des charges et je l'aurais volontiers accomplie mais elle commence à me prendre beaucoup de temps de recherche et le projet touchera bientôt à sa fin. Philippe est d'accord de s'en occuper. Il a appelé le helpdesk de l'état du Valais pour savoir où en était la demande que je leur avais soumise Lundi 18 mars. La procédure est en cours. Nous avons déjà configuré l'envoi vers le relai SMTP en éditant le fichier main.cf de PostFix. Nous attendons de leur nouvelles.

Objectifs pour le prochain jour

Effectuer une série de tests solides sur les modifications apportées aujourd'hui sur le module d'envoi des fichiers pour s'assurer qu'il n'y ait pas de bugs bloquants. Effectuer des corrections sur l'envoi des mails lors de l'expiration des dossiers. J'ai remarqué qu'il y avait plusieurs bugs : Dans certains mails, l'encodage n'est pas bon où le code html est affiché en toute lettres (dans le sujet ou corps de message). L'heure exacte de l'expiration du dossier n'est pas la bonne.

Au niveau de la planification, je suis en train d'apporter toutes les corrections et améliorations possibles dans le temps qu'il me reste. Tout ce qui avait été planifié est effectué, le cahier des charges est donc rempli à l'exception de la navigation sécurisée que je dois encore travailler. Sur le serveur de production, le certificat du domaine vs.ch est utilisé, ceci n'était pas prévu puisque cette tâche était comprise dans mon cahier des charges. Je l'effectuerai tout de même comme il se doit en générant mon propre certificat sur le poste de développement ainsi que sur le serveur de production pour un accès en interne.

Tâches réalisées

2 Réalisation
2.3 Modules spécifiques
2.3.3 Transfert de fichier — 100%
2.3.7 Navigation sécurisée — 20%

Vendredi, 22 Mars 2013

Février
Lundi 25
Jeudi 28

Mars
Vendredi 01
Lundi 04
Jeudi 07
Vendredi 08
Lundi 11
Jeudi 14
Vendredi 15
Lundi 18
Jeudi 21
Vendredi 22
Lundi 25
Mardi 26
Mercredi 27

Périodes		Déroulement	
		Tâches effectuées	Remarques sur les tâches effectuées
07:30	09:00	Modification de la requête d'envoi des mails d'avertissement avant la suppression des dossiers	Je me suis rendu compte que ma requête n'était pas complète car elle sélectionnait aussi les dossiers expirant au moment même… Ce qui faisait que je recevais le mail une première fois un jour avant l'expiration puis une deuxième fois en même temps que le mail de suppression des fichiers. J'ai corrigé cela. J'ai aussi créé une méthode pour convertir les entités HTML en caractères mais je n'ai pas encore réussi à la faire fonctionner, surtout parce que dans la base de données mes textes ne sont pas encore en entités, je vais travailler là-dessus. Par contre j'ai remarqué que j'avais oublié quelque chose : lorsqu'on glisse des fichiers dans la zone de drag&drop, ils ne sont pas renommés !
09:00	10:30	La méthode de conversion des entités en caractère n'est pas bonne, elle entrainerait des doublons dans la base de données et ne marche pas…	La méthode dont j'ai parlé ci-dessous ne fonctionnais pas car les entités étaient elles aussi mal encodées. J'ai donc cherché un moyen de résoudre ce problème autrement. J'ai réussi à modifier l'encodage de la base de données mais ça ne fonctionne toujours pas !
10:30	11:30	Encodage des mails	Après beaucoup de recherche et de tests, j'ai réussi à résoudre le problème de l'encodage. Je me suis empressé de l'écrire dans ma documentation technique car j'ai peiné depuis ce matin sur ce problème ! Il fallait également renseigner l'encodage utilisé dans la méthode d'envoi des mails en Java.
11:30	13:00	Vérification de l'existence des fichiers Affichage des fichiers déjà téléchargés	Lorsqu'un fichier n'existe pas (pour cause d'erreur ou que sais-je…) le lien n'est pas activé, un message indique que ce fichier n'existe pas. J'ai tenté d'afficher par qui les fichiers avaient déjà été téléchargés lorsqu'on passait sur un fichier du dossier mais je n'arrive pas à afficher, le tableau de valeur est chargé mais l'affichage n'est pas mis à jour… Je vais réfléchir à une meilleure façon de présenter ces données…
13:00	14:00	Pause de midi	
14:00	15:30	Réaffichage des destinataires	J'ai affichés différemment les composants qui indiquent quels contacts sont autorisés dans le

		dossier. Ils se présentent maintenant sous forme de boutons. J'avais besoin de ce composant pour pouvoir lier un évènement. Avant ça je n'affichais que du texte en HTML et je n'avais donc aucun moyen de pouvoir détecter le clic sur le texte. J'affiche une icône si le contact a des actions journalisées, sinon, je n'affiche rien et si le bouton est cliqué, un message indique que le contact n'a pas encore consulté le dossier.
15:30 17:00	Mise en forme et intégration des informations dans les tableaux	Deux tableaux sont affichés, l'un avec le résumé des actions effectuées par le contact et l'autre avec toutes les actions détaillées. J'ai eu un problème lorsque la fenêtre était fermée puis ré-ouverte. Le tableau n'était plus chargé de données. J'avais oublié de repartir de la ligne 0 dans mon DataSet.

Problèmes rencontrés & résolution

Beaucoup de problèmes au niveau de l'encodage des caractères aujourd'hui. Ce problème se produisait uniquement dans les mails, sauf que ce soir, en regardant l'application en ligne depuis la maison, j'ai remarqué que tous les caractères accentués étaient affichés de la sorte : CrÃ©ation.
Je comprends le problème mais je ne peux malheureusement pas le résoudre depuis la maison, j'attendrai donc Lundi matin pour le faire. Le problème vient du fait que j'ai modifié l'encodage par défaut de la base de données. Ma base de données est réellement en UTF-8 et les mails sont envoyés en latin1. Le site est affiché en UTF-8, mais j'ai donné l'ordre à la base de données de tout gérer en latin1… Il va donc falloir que je gère tout en UTF-8. La base de données, l'interface et l'envoi de mails. Ce fut le principal problème de la journée qui m'a fait perdre beaucoup de temps.

Objectifs pour le prochain jour

Corriger les problèmes d'encodage, Il faut que je trouve un compromis et que j'utilise le même encodage partout pour éviter ces problèmes. Je devrai probablement revenir à la configuration de base de MySQL dans un premier temps, pour éviter de m'embrouiller. Je vais essayer d'envoyer les mails en UTF-8 et voir ce que ça donne. Je vais également essayer de terminer l'installation et l'intégration dans Tomcat du certificat SSL. Je vais commencer à rassembler, trier, contrôler les documentations pour éviter d'oublier un point important. Je dois encore intégrer les schémas UML correspondants aux classes.

Tâches réalisées

2 Réalisation
2.3 Modules spécifiques
2.3.6 Suppression automatique des fichiers 95%
2.3.9 Autres (Visualisation des journalisations) 90%

Lundi, 25 Mars 2013

Périodes		Déroulement	
		Tâches effectuées	Remarques sur les tâches effectuées
07:30	08:30	Résolution du problème d'encodage dans les mails	J'ai supprimé toutes les configurations que j'avais effectuées vendredi pour l'encodage des mails en latin1. Comme la base de données est en UTF-8, j'ai décidé de ne pas me compliquer la vie, je redirige tout en UTF-8 : interface web, base de données, mails, … J'ai résolu le problème de cette manière. Il y a deux paramètres important à ne pas négliger, c'est la conversion des bytes du message en UTF-8 (Corps et sujet du message !) et l'indication au programme de l'encodage utilisé pour l'envoi.
08:30	09:30	Résolution d'un problème de requête SQL pour la suppression des fichiers et l'expiration des dossiers. Résolution d'un bug d'affichage	La requête n'est pas bonne, elle spam les gens car le jour de la suppression, la personne reçoit chaque heure le mail de rappel avant l'expiration du dossier. J'ai réussi à résoudre ce problème. Il y avait également un bug : lors de la connexion avec un utilisateur, la déconnexion, puis la reconnexion avec un autre utilisateur, la liste des contacts de l'utilisateur précédent restait affichée car elle était déclarée avant l'initialisation du menu… J'ai résolu ce bug en l'initialisant dans le constructeur.
09:30	10:30	Ajout d'un champ à la base de données pour prévoir des utilisateurs « administrateurs »	Affichage d'un onglet « Administration » dans le menu accordéon, uniquement si l'utilisateur est détecté comme administrateur. Pour l'instant il n'y a encore rien d'affiché, uniquement l'onglet sans aucun contenu. Je vérifie qu'un simple utilisateur (sans droits d'administration) ne voie pas cet onglet et que la connexion et inscription reste toujours possible après avoir ajouté ce champ à la table des utilisateurs.
10:30	12:00	Création de la page de gestion des utilisateurs pour la « zone admin »	Dans la zone d'administration, j'ai créé la page qui répertorie les utilisateurs et qui devra permettre de les assigner en administrateur, de valider les comptes, de pouvoir les supprimer, … Mais je n'ai pas encore implémenté ces actions. Je n'ai que travaillé l'affichage pour le moment.
12:00	13:00	Navigation sécurisée	Auto-génération d'un nouveau certificat SSL pour l'utilisation en locale. Inclusion dans la configuration Tomcat. J'arrive à accéder avec le port 8443 mais pas encore avec https, il doit me manquer une ligne dans le connecteur. J'aimerais aussi que dès qu'on accède, on soit directement redirigé vers le protocole HTTPS.
13:00	14:00	Pause de midi	
14:00	16:00	Création de la page pour la visualisation des journalisations dans la zone d'administration	Cette tâche est la dernière de la 2e partie du cahier des charges. Je n'affiche par contre pas de graphique car je ne vois quelles données utiles je pourrais afficher sous forme de graphique, un tableau est plus parlant à mon avis…

16:00	17:00	Formatage des données du tableau d'affichage des journalisations et tests sur le serveur	Lorsqu'on voyait le tableau, il ne donnait pas assez les informations du premier coup d'œil. Il n'est pas assez intuitif car il regroupe beaucoup trop d'informations textuelles. Pour le rendre plus attractif, j'ai cherché un moyen de générer de nouvelles colonnes où j'affiche une image à la place du texte. L'alliage des deux moyens de communication est beaucoup plus clair. Ceci étant fait, j'ai réglé quelques problèmes liés à la traduction (les textes des différents menus contextuels ne se traduisaient pas car je les déclarais dans l'entête de la classe avant que la langue soit détectée par l'application. J'ai donc déplacé toutes les actions des menus contextuels dans les constructeurs des différentes classes. Pour terminer, j'ai testé le tout sur le serveur de production.

Problèmes rencontrés & résolution

Je n'ai pas rencontré de problèmes particuliers aujourd'hui. J'ai résolu tous ceux que j'ai pu encore apercevoir mais il me semble que la partie 1 du cahier des charges (partie obligatoire), est bien terminée. Il y a toujours quelques erreurs qui surviennent parfois dans des situations vraiment particulières que je n'avais jamais rencontrées, mais quand cela arrive, je m'empresse de les corriger. J'ai attaqué pleinement la partie facultative du cahier des charges, je ne pense pas qu'elle pourra être entièrement terminée mais l'essentiel pour moi est d'avoir terminé la partie obligatoire, de la documenter au mieux, pour mettre toutes les chances de mon côté. Aucun problème particulier à signaler.

Objectifs pour le prochain jour

Terminer proprement les tests et rapports de tests. Préparer toutes les documentations, définir leur emplacement, tout préparer pour l'impression que je lancerai très tôt mercredi matin car il y a vraiment beaucoup de pages et je dois faire attention de ne rien oublier. Terminer si j'arrive le module d'administration pour la visualisation, et dans l'idéal, j'aimerai beaucoup déjà pouvoir créer un module de gestion des paramètres et des traductions. Mais je ne pense pas avoir le temps de faire tout cela en un seul jour.

Tâches réalisées

2 Réalisation

2.3 Modules spécifiques

- 2.3.2 Journalisation des opérations — 100%
- 2.3.4 Envoi d'URL cryptées et extraction des données liées — 100%
- 2.3.6 Suppression automatique des fichiers — 100%
- 2.3.7 Navigation sécurisée — 100%
- 2.3.9 Autres (Zone d'administration) — 60%

Mardi, 26 Mars 2013

Février
Lundi 25
Jeudi 28

Mars
Vendredi 01
Lundi 04
Jeudi 07
Vendredi 08
Lundi 11
Jeudi 14
Vendredi 15
Lundi 18
Jeudi 21
Vendredi 22
Lundi 25
Mardi 26
Mercredi 27

Périodes		Déroulement	
		Tâches effectuées	Remarques sur les tâches effectuées
07:30	09:00	Tests et rédaction du rapport en fonction	Corrections des bugs où des oublis en fonction du résultat des tests
09:00	10:30	Administration (Journalisations)	Gestion de l'affichage des journalisations en fonction des catégories et des filtres. Je n'ai pour l'instant implémenté que l'affichage des journalisations de connexion et de validation des comptes. Je dois maintenant adapter le tableau pour pouvoir afficher également des journalisations de téléchargement.
10:30	11:30	Correction de bugs, ajout de traductions manquantes et traduction des mots en allemand	J'ai détecté plusieurs bugs que j'ai résolus. Ceux-ci étaient liés au déplacement des actions des menus contextuels dans le constructeur. Certaines variables avaient été supprimées. Il n'y avait pas d'erreur puisque la classe parente avait les actions en *protected* mais les actions n'étaient pas exécutées lors du clic. J'ai également ajouté tous les mots des pages d'administration dans la base de données pour traduction et j'ai traduit quelques mots en allemand (avec Google traduction)
11:30	13:00	Résolution de plusieurs bugs que je suis content d'avoir trouvé !	Il y avait particulièrement deux bugs que je suis content d'avoir pu résoudre. Le premier survenait lorsqu'on envoyait plusieurs fichiers non autorisés, ils s'ajoutaient dans un tableau java et s'enregistraient quand même lorsque je cliquais sur terminé. Du coup, les fichiers autorisés n'étaient pas enregistrés dans la base de données et tous les fichiers du dossier étaient inexistants. J'ai également ajouté un contrôle que j'avais oublié : Lorsque l'utilisateur renommait des fichiers, il pouvait mettre des noms vides… Maintenant, s'il le fait, le nom original est conservé.
13:00	14:00	Pause de midi	
14:00	15:30	Mise à jour et contrôle du rapport de test	J'ai mis à jour le rapport de test final en ajoutant déjà quelques nouveaux tests effectués et quelques autres que j'avais oublié. Je n'ai pas encore terminé car je rédige maintenant

			proprement les tests pour toute la partie concernant le module d'administration qui est tout neuf. Je pense que la programmation s'arrête ici pour mon projet. Je vais travailler jusqu'à la fin pour la préparation et la révision de mes documentations.
15:30	16:30	Test général et final de l'application. Préparation des documentations, génération des PDF des documentations et des codes personnels.	
16:30	17:30	Impression des rapports de tests, du PV, du manuel utilisateur et de quelques parties de codes.	L'imprimante m'indique un bourrage papier. Pourtant j'ai ouvert toutes les portes il n'y a aucun papier à l'intérieur. J'espère que demain tout se passera bien parce que j'ai peur qu'il y ait une panne. En tout cas, mes documentations sont prêtes, je n'ai plus qu'à les imprimer.

Problèmes rencontrés & résolution

Aujourd'hui j'ai cherché le plus possible les problèmes qui pourraient survenir et tous ceux que j'ai trouvé je les ai résolus. Ce n'était que des petits problèmes qui survenaient dans des cas bien particulier, je les ai donc uniquement notifiés dans ce journal de bord. J'espère qu'il n'y en aura plus… Je sais qu'il n'existe pas d'application parfaite et qu'il y aura toujours quelques bugs mais en tout cas, à ce stade tout me semble fonctionner correctement.

Objectifs pour le prochain jour

Imprimer toute la documentation en trois exemplaires, la relier et l'envoyer aux experts.

Tâches réalisées

2 Réalisation
2.3 Modules spécifiques
2.3.8 Rapport de test 100%
2.3.9 Autres (Zone d'administration) 100%

Mercredi, 27 Mars 2013

Février
Lundi 25
Jeudi 28

Mars
Vendredi 01
Lundi 04
Jeudi 07
Vendredi 08
Lundi 11
Jeudi 14
Vendredi 15
Lundi 18
Jeudi 21
Vendredi 22
Lundi 25
Mardi 26
Mercredi 27

Périodes		Déroulement	
		Tâches effectuées	**Remarques sur les tâches effectuées**
07:00	09:00	Documentations : derniers contrôles, amélioration de la mise en page, mise à jour des index et références, impression des documents	Documents prêts : JavaDoc Code source du projet Documentation technique
09:00	10:30	Complétion du rapport de projet	Rédaction de la conclusion du rapport de projet, inclusion de la première page du cahier des charges tel qu'il a été signé au départ et du procès-verbal. Génération du rapport au format PDF et vérification que toutes les informations soient présentes et positionnées correctement sur les pages.
10:30	11:00	Impression du rapport de projet en 3 exemplaires	
11:00	11:30	Révision des dernières pages du journal de bord avant l'impression.	
11:30	12:00	Impression du journal de bord et organisation des différentes documentations. (assemblage pour les 3 exemplaires)	
12:00	13:00	Pause de midi	
13:00	14:30	Reliage des documents chez Felix bureautique.	
14:30	15:30	Préparation des documentations pour l'envoi par la Poste.	
15:30		Envoi des documentations aux deux experts.	

Conclusion

Le projet se termine bien, je suis satisfait des rapports et du résultat de l'application. Je pensais rencontrer des problèmes liés aux très nombreuses pages qui constituent mes documentations mais j'ai finalement réussi à diviser par 4 le nombre de pages.

CCCVs-Transfert

Documentation technique

Caisse de compensation du canton du Valais
Ausgleichskasse Wallis

Service informatique

Dominique Roduit

27/03/2013

dominique.roduit@avs.vs.ch

TABLE DES MATIERES

1. INTRODUCTION

Ce document a pour but de détailler, analyser, clarifier les informations techniques relatives au projet, tel que les champs de la base de données, la navigation dans l'interface WEB, la structure de l'interface, l'envoi des fichiers ou encore d'expliciter les erreurs rencontrées au cours de la réalisation avec des solutions détaillées.

Les images illustratives utilisées proviennent du site iconfinder.com et sont libres d'utilisation.

2. VAADIN

2.1. INSTALLATION

L'installation de la librairie Vaadin et de tous les outils nécessaires à la création de projets, de thèmes, de widgets et de composants Vaadin se fait très simplement depuis le menu "Help > Eclipse Marketplace" dans lequel il suffit de taper "Vaadin" et de cliquer sur "Install".

2.2. VERSION

Version de Vaadin installée pour ce projet : ***6.8.8.***

Vaadin évolue continuellement, la dernière version publiée à ce jour est la ***7.0.0***. Cette nouvelle version publiée le 4 Février 2013 apporte d'importantes nouveautés et mises à jour. De nombreux composants y sont ajoutés ou améliorés. De nouveaux thèmes sont également disponibles.

Les différentes raisons pour lesquelles je n'ai pas opté pour cette version sont les suivantes :

- ✓ La CCCVs m'a permis de me former pour Vaadin 6 en m'autorisant à suivre des cours avec un expert Vaadin durant 3 jours.
- ✓ Actuellement, aucun développeur Vaadin de la CCCVs n'a déjà migré sur la version 7. Celle-ci change considérablement sa manière d'utiliser les thèmes et le code utilisé dans les applications Vaadin 6 n'est pas entièrement compatible avec la version 7. Si je développe en Vaadin 7, ils n'auront donc pas forcément déjà les connaissances pour s'occuper de mon application en cas de problème lorsque je ne serai plus là.
- ✓ Certains "add-on" dont j'ai un réel besoin pour mon application ne sont pas compatibles avec Vaadin 7, notamment l'add-on "EasyUploads" qui est un des composant clé pour la bonne réalisation de ce projet.

Voici une brève explication à propos des versions de vaadin :

2.2.1. FIXPACK

Lorsque nous changeons de Fixpack, Vaadin garantit que tout fonctionne exactement de la même manière. Il n'y a aucun changement de code à faire. Le Fixpack corrige des bugs mineurs.

2.2.2. RELEASE

Un changement de Release signifie une mise à jour plus conséquente. Il peut y avoir de nouveaux composants. Il est conseillé de re-tester l'application lorsqu'on change de release.

2.2.3. VERSION

Un changement de version implique de grosses et nombreuses modifications. Il peut y avoir un changement radical dans la manière d'utiliser les thèmes et composants. Beaucoup de nouveaux composants y sont ajoutés. La compatibilité avec la version antérieure n'est absolument pas garantie.

2.3. CRÉATION DU PROJET VAADIN

A l'installation de Vaadin depuis l''eclipse marketplace'', un plugin est ajouté automatiquement. Ce plugin permet de créer des nouveaux projets Vaadin, des thèmes personnalisés ou des widgets.

Pour ce faire, il suffit de se rendre dans le menu "File > New > Other… > Vaadin > Vaadin 6 Project".

- Nom du projet : CCCVsTransfert
- Target runtime : Apache Tomcat v7.0
- Configuration : Vaadin 6, Java 6, Servlet 3.0
- Vaadin version : 6.8.8

2.4. THÈME PERSONNALISÉ

Vaadin 6 offre 3 thèmes natifs différents.

- ✓ Thème par défaut
- ✓ Runo
- ✓ Reindeer

Ces thèmes contiennent toutes les images et feuilles de styles nécessaires pour tous les composants Vaadin de notre application, cependant un des point du cahier des charges est de créer un design moderne, novateur, sobre et intégré à la charte graphique de la Caisse de compensation. Ceci implique forcément la création d'un thème Vaadin personnalisé. Je me base sur un des 3 thèmes et ajoute ma propre feuille de style CSS par-dessus.

Voici la démarche à réaliser pour créer son propre thème Vaadin :

- Dans le dossier WebContent du projet, créer un nouveau dossier "VAADIN".
- Dans ce dossier, y ajouter un dossier "themes"

- Dans le dossier "themes", créer un dossier portant le nom du nouveau thème, dans mon cas "CccvsDesign".
- Dans le dossier "CccvsDesign", créer une nouvelle feuille de style CSS nommé "styles.css" et un nouveau dossier nommé "img".
- Au début de la méthode init() de l'application, indiquer le code

```
setTheme("CccvsDesign");
```

Il suffit maintenant de rafraîchir le projet dans l'espace de travail Eclipe et le nouveau thème est pris en compte. Chaque fois qu'on ajoute une image dans le dossier "img", il est important de rafraîchir le projet, sinon il y aura une erreur de ce type, indiquant que la ressource n'a pas été trouvée :

```
INFO: Requested resource [VAADIN/themes/CccvsDesign/img/collapsed.png] not found
from filesystem or through class loader.
```

2.4.1. ICÔNE PERSONNALISÉE

Par défaut, c'est le logo de Vaadin qui est utilisé comme icône dans l'onglet du navigateur.

Pour la modifier, il suffit d'insérer l'icône personnalisée nommée "favicon.ico" dans le dossier du thème utilisé (CccvsDesign).

2.4.2. MODIFICATION DES MESSAGES SYSTÈMES

Vaadin intègre nativement des messages systèmes. Parmi ces derniers, on peut trouver :

Il en existe plusieurs autres.

Par défaut, ces messages sont en anglais, mais heureusement, ils sont configurables. Pour modifier le contenu des messages système, il faut ajouter la méthode suivante dans la classe exécutée au lancement de l'application (*CccvsTransfert*) :

```
public static SystemMessages getSystemMessages() {
    CustomizedSystemMessages messages = new CustomizedSystemMessages();
    messages.setSessionExpiredCaption("Titre de l'erreur");
    messages.setSessionExpiredMessage("Description de l'erreur");

}
```

Tous les messages d'erreurs personnalisés doivent être ajoutés dans cette méthode.

3. QUELQUES DÉFINITIONS IMPORTANTES

Souvent, des termes tels que « Utilisateur interne » apparaîtront, notamment dans les commentaires. Pour comprendre comment fonctionne l'application, il est important d'être au clair sur certains mots clés souvent utilisés.

Utilisateur/contact interne

Une personne « interne » est un individu travaillant à la Caisse de compensation de Sion. Une telle personne est identifiable sur l'application grâce au format de son adresse e-mail qui est utilisé pour authentifier les utilisateurs. L'adresse e-mail d'une personne interne se termine par '@avs.vs.ch'.

Ruban / Ribbon

L'application comporte deux rubans.

- Le premier est la partie supérieure du menu qui contient le nom et la version de l'application. C'est une zone non dynamique, où le contenu ne change pas selon les pages. Toutefois, la taille du menu peut varier car la page de téléchargement affiche les informations du dossier dans la partie latérale gauche.
- Le deuxième ruban est celui du corps de page qui contient les boutons d'action. Pour mieux comprendre ce concept, veuillez vous référer à la section traitant du design de l'application et des différents layout. *(Chapitre 8.1. - p.17)*

4. CONFIGURATION GÉNÉRALE TOMCAT

Après quelques déploiement à chaud de l'application, une erreur PermGen space provoque le blocage du serveur d'application. Après une telle erreur il ne reste plus qu'à stopper le processus du serveur puis à le relancer.

```
java.lang.OutOfMemoryError: PermGen space
   at java.lang.ClassLoader.defineClass1(Native Method)
   at java.lang.ClassLoader.defineClass(ClassLoader.java:620)
   at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:124)
```

Il semble que cette erreur soit liée à l'utilisation de bibliothèques externes qui chargent dynamiquement les classes. Celles-ci ne sont pas compatibles avec le redéploiement car l'URLClassLoader de Tomcat garde des références sur des objets dont les classes ont été chargées dynamiquement lors de l'exécution, ce qui empêche le garbage collector de faire correctement son travail.

Le PermGen space est une zone mémoire qui contient tout ce qui n'est pas géré par le garbage collector ; tout ce qui est relatif aux classes (leur structure : méthodes, champs, annotations…). Plus on a de classes différentes, plus il faudra augmenter la taille de cette zone mémoire.

Voici les paramètres à modifier pour éviter ce problème :

Tomcat sur Windows (Poste de développement)

Ajouter ces deux lignes au bas du fichier conf/catalina.properties.

```
-XX:MaxPermSize=256M
-XX:PermSize=256M
```

Tomcat sur Linux (Serveur de production)

Modifier les mêmes lignes qu'au-dessus, dans le fichier /opt/tomcat7/bin/catalina.sh

Sources d'aide :

Résolution de l'erreur PermGen : http://bit.ly/16MWSM
Raisons de l'erreur PermGen : http://bit.ly/aUbCg1

5. BASE DE DONNÉES

5.1. DÉFINITIONS

PK	➔	Clé primaire	Primary key
NN	➔	Non null	Not null
AI	➔	Auto-incrémentation	Auto increment
FK	➔	Clé étrangère	Foreign Key

5.2. CONNEXION

5.2.1. PRÉREQUIS

Pour la connexion à la base de données MySQL en Java, il est tout d'abord nécessaire d'ajouter les drivers JDBC (Java Database Connectivity) qui permettent d'effectuer le lien entre le programme Java et un grand nombre de base de données. Ceux-ci sont se présentent sous la forme d'une librairie Java à ajouter dans le "build path" ou directement dans le dossier WebContent/WEB-INF/lib de l'application.

Drivers pour la connexion MySQL : Connector/J *(mysql-connector-java-5.1.23-bin.jar)*

5.2.2. ETABLISSEMENT DE LA CONNEXION

1. Importer les packages SQL

```
import java.sql.*;
```

2. Charger le pilote

```
try {
	Class.forName("com.mysql.jdbc.Driver").newInstance();
} catch (Exception ex) {
}
```

3. Etablir la connexion

```
try {
	Connection conn = DriverManager.getConnection(
	"jdbc:mysql://localhost/db_cccvstransfert", "cccvstransfert",
"K*******");
} catch(SQLException ex) {
}
```

5.3. CRÉATION DES UTILISATEURS

Pour des questions de sécurité, lorsqu'une application doit être disponible sur internet, il faut si possible éviter de s'authentifier en tant qu'utilisateur "root" depuis notre application. Il est préférable de créer un utilisateur en lui restreignant les droits au niveau des requêtes d'administration de la base de données. Voici l'utilisateur créé pour cette application.

Je crée tout d'abord un utilisateur avec lequel je peux me connecter depuis n'importe où, pour pouvoir tester déjà si la connexion est possible.

```
CREATE USER 'cccvstransfert'@'%' IDENTIFIED BY 'K*******';

GRANT ALL PRIVILEGES ON * . * TO 'cccvstransfert'@'%' IDENTIFIED BY 'K*******' WITH
GRANT OPTION MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR
0 MAX_USER_CONNECTIONS 0;
GRANT ALL PRIVILEGES ON `db_cccvstransfert` . * TO 'cccvstransfert'@'%';
```

Ce n'est qu'une fois la connexion établie avec succès que je crée un utilisateur avec lequel je peux me connecter seulement depuis certains endroits.

```
# Création de l'utilisateur
CREATE USER 'cccvstransfert'@'localhost' IDENTIFIED BY 'K*******';
# Privilèges globaux : aucun droit
GRANT USAGE ON *.* TO 'cccvstransfert'@'localhost'
IDENTIFIED BY 'K*******' WITH
MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0
MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;
# Privilèges sur la base de donnée pour cette utilisateur
GRANT ALL PRIVILEGES ON `db_cccvstransfert` . * TO ' cccvstransfert
'@'localhost';
```

Copier le même code en remplaçant 'localhost' par '10.76.211.191'. L'adresse IP mentionnée étant celle de mon PC, pour que je puisse également y accéder depuis mon poste de développement.

Voici comment se présente la table des utilisateurs à la fin de l'opération :

```
prdWTransfert:/usr/bin # mysql> select User,host from user;
+----------------+---------------+
| cccvstransfert | 10.76.211.191 |
| root           | 127.0.0.1     |
| root           | ::1           |
| cccvstransfert | localhost     |
| root           | localhost     |
| root           | prdWTransfert |
+----------------+---------------+
```

5.3.1. IMPORTATION DU SCRIPT SQL SUR LE SERVEUR

Le script SQL contenant le code de construction de la base de données ainsi que toutes ses tables doit être ajouté préalablement sur le partage \\pcdata\work avant de pouvoir être atteignable depuis le serveur. Pour rappel, pour monter un partage sur le serveur, il faut exécuter la commande suivante :

```
mount cccapp:/work /work
```

Voici maintenant la commande permettant d'importer le script SQL sur le serveur

```
mysql -u cccvstransfert -p db_cccvstransfert < /work/TPI/cccvstransfert.sql
```

5.4. TABLES

5.4.1. UTILISATEURS DE L'INTERFACE WEB

Nom de la table : **trans_users**

Table qui va contenir les utilisateurs de l'interface Web.

NOM	TYPE	PK	NN	AI	FK	Commentaire
PKNoUser	int(11)	×	×	×		Clé primaire
user_mail	varchar(255)		×			Adresse e-mail de l'utilisateur
user_pass	varchar(260)		×			Mot de passe de l'utilisateur (SHA256)
user_internal	bool		×			true : l'utilisateur travaille à la CCCVs false : L'utilisateur vient de l'extérieur
user_validation	bool		×			true : L'adresse e-mail est validée false : La validation est en attente
user_validation_date	datetime		×			Date à laquelle l'utilisateur a validé son adresse e-mail
user_langue	varchar(2)		×			Langue préférée de l'utilisateur
user_admin	bool					Indique si l'utilisateur est administrateur

5.4.1. TRADUCTIONS DES TEXTES DE L'INTERFACE WEB

Nom de la table : trans_translate

Table qui va contenir tous les textes et contenus traduits de l'application.

NOM	TYPE	PK	NN	AI	FK	Commentaire
PKNoTranslate	int(11)	×	×	×		Clé primaire
trans_label	varchar(45)		×			Mot clé pour un texte
trans_fr	text					Texte en français
trans_de	text					Texte en allemand

5.4.1. DESTINATAIRES POUR L'ENVOI D'UN DOSSIER

Nom de la table : trans_recipients

Table qui va contenir les contacts auxquels un dossier a été envoyé.

NOM	TYPE	PK	NN	AI	FK	Commentaire
PKNoRecipient	int(11)	×	×	×		Clé primaire
FKNoFolder	int(11)		×		×	Lien vers le dossier de transfert
FKNoContact	int(11)		×		×	Lien vers le destinataire du transfert

5.4.1. DOSSIERS CONTENEURS DES FICHIERS TRANSFÉRÉS

Nom de la table : trans_folders

Table qui va contenir les dossiers créés par les utilisateurs. Les dossiers sont des conteneurs des fichiers envoyés par l'utilisateur.

NOM	TYPE	PK	NN	AI	FK	Commentaire
PKNoFolder	int(11)	×	×	×		Clé primaire
FKNoUser	int(11)		×		×	Assignation du dossier à un utilisateur de l'application
folder_name	varchar(50)		×			Nom du dossier de transfert
folder_description	text					Description du dossier
folder_creation_date	datetime		×			Date de création du dossier
folder_expiration	datetime		×			Date de destruction du dossier
folder_archive	bool		×			true : le fichier est archivé false : le dossier est toujours valable

5.4.1. FICHIERS D'UN TRANSFERT

Nom de la table : trans_files

Table qui va contenir les fichiers envoyés par les utilisateurs.

NOM	TYPE	PK	NN	AI	FK	Commentaire
PKNoFile	int(11)	×	×	×		Clé primaire
FKNoFolder	int(11)		×		×	Assignation du fichier à un dossier de transfert
file_name	varchar(150)		×			Nom du fichier tel qu'il est enregistré sur le disque (sans le chemin)
file_rename	varchar(70)					Nom du fichier spécifié par l'utilisateur (après renommage)
file_size	bigint(20)		×			Taille du fichier (en octets)
file_extension	varchar(12)		×			Extension du fichier
file_description	text					Description du fichier spécifiée par l'utilisateur

5.4.1. Liste des contacts d'un utilisateur

Nom de la table : trans_files

Table qui va contenir les contacts créés par les utilisateurs.

NOM	TYPE	PK	NN	AI	FK	Commentaire
PKNoContact	int(11)	×	×	×		Clé primaire
FKNoUser	int(11)		×		×	Assignation du contact à un utilisateur de l'application
contact_name	varchar(25)		×			Nom ou description du contact
contact_mail	varchar(255)		×			Adresse e-mail du contact
contact_creation_date	datetime		×			Date de création du contact
contact_internal	bool		×			true : Contact interne à la CCCVs false : Contact externe

5.4.1. Paramètres de l'application

Nom de la table : trans_app_param

Table qui va contenir les paramètres importants de l'application. Par ex. le nom et la version.

NOM	TYPE	PK	NN	AI	FK	Commentaire
PKNoParam	int(11)	×	×	×		Clé primaire
parm_key	varchar(45)		×			Clé/Nom du paramètre (en majuscule)
parm_value	text		×			Valeur du paramètre
parm_description	text					Description du paramètre

5.4.1. Journalisation des téléchargements et consultations

Nom de la table : trans_journal_folders

Table qui va contenir les actions effectuées par les destinataires d'un transfert sur un dossier.

NOM	TYPE	PK	NN	AI	FK	Commentaire
PKNoJourFolder	int(11)	×	×	×		Clé primaire
FKNoContact	int(11)		×		×	Liaison vers le contact qui à téléchargé le fichier (et non l'auteur du dossier !)
FKNoFolder	int(11)		×		×	Liaison vers le dossier contenant le fichier
FKNoFile	int(11)				×	Liaison vers le fichier téléchargé
joufo_action	varchar(60)		×			Action effectuée (download, consult)
joufo_date	datetime		×			Date de l'action

5.4.1. Journalisation des actions lors de la connexion

Nom de la table : trans_journal_connections

Table qui va contenir les actions effectuées par les utilisateurs lors de la connexion.

NOM	TYPE	PK	NN	AI	FK	Commentaire
PKNoJourConnection	int(11)	×	×	×		Clé primaire
joco_mail	varchar(255)		×			Adresse e-mail qui tente la connexion
joco_date	datetime		×			Date de l'action
joco_ip	varchar(15)		×			Adresse IP publique de la personne qui se connecte
joco_os	varchar(45)		×			Système d'exploitation de la personne qui se connecte
joco_browser	varchar(45)		×			Navigateur de la personne qui se connecte
joco_action	varchar(100)		×			Action effectuée, erreur générée, …
joco_comment	text		×			Détails sur l'action

6. PARAMÉTRAGE DE L'APPLICATION

parm_key Clé/Nom du paramètre	parm_value Valeur du paramètre	parm_description Description du paramètre
APP_VERSION	1.0.0	Version de l'application
APP_NAME	CCCVs-Transfert	Nom de l'application
EMAIL_INTERNE_SUFFIXE	avs.vs.ch	Suffixe des adresses e-mail considérées comme inte...
UPLOAD_MAX_SIZE	1073741824	Taille maximal d'un transfert de fichier, en Octet...
FOLDER_MAX_AVAILABILITY	10	Disponibilité maximum des dossiers, en jours.
APP_MAIL_FROM_NAME	CCCVs-Transfert	Nom affiché lors de l'envoi d'un email.
APP_MAIL_FROM	support@avs.vs.ch	Adresse mail utilisée pour envoyer des messages au...
APP_DEFAULT_LANGUAGE	fr	Langue par défaut de l'application
PASS_MIN_LENGTH	3	Longueur minimum d'un mot de passe utilisateur
PASS_MAX_LENGTH	50	Longueur maximum d'un mot de passe utilisateur
LAYOUT_CONNEXION_HEADER_HEIGHT	64px	Hateur du header pour le layout de connexion
LAYOUT_MAIN_HEADER_HEIGHT	54px	Hauteur du header pour le layout principal
LAYOUT_MAIN_RIBBON_HEIGHT	50px	Hauteur du ruban pour le layout principal

Un des objectifs qui ne figure pas clairement dans le cahier des charges, mais qui fait partie intégrante de la bonne réalisation de l'application, est le paramétrage de cette dernière. D'après mon interprétation, toutes les valeurs utilisées plus de deux fois dans l'application doivent devenir paramètres. L'avantage de stocker des paramètres dans la base de données et de pouvoir à tout moment modifier une valeur qui sera mise à jour sur l'interface entière sans avoir à recompiler les sources de l'application.

6.1. RECONNAISSANCE AUTOMATIQUE ET CONVERSION DES PARAMÈTRES EN CHAINES

L'application est écrite de sorte à reconnaître automatiquement les paramètres entrés directement dans les textes de la base de données. Je m'explique :

La page « À propos » affiche le nom et la version de l'application. Le texte visible vient, comme tout le contenu de l'application, de la base de données.

Imaginons que le texte de la page provient d'une entrée de la table des traductions nommée « CONTENT ».

Une table a été créée uniquement dans le but de conserver les paramètres importants de l'application. Ces paramètres doivent être utilisés pour que l'application soit la plus souple possible.

Le champ qui contient la valeur d'un paramètre peut encapsuler d'autres paramètres de la base de données, voici une description pour éclaircir mes propos :

Si on écrit en clair dans le champ « CONTENT » le nom et la version de l'application, lorsque les paramètres « nom » et « version » également enregistrés dans la table seront modifiés, le nom et la version écris en clair dans le champ « CONTENT » ne seront pas modifiés car ils ne feront pas référence aux paramètres.

Pour pouvoir utiliser des paramètres dans n'importe quel texte écrit en clair dans la base de données, une méthode « ***parmConverter*** » a été implémentée dans la class « ***Utilities*** ».

Il faut alors remplacer les textes par les paramètres correspondant. Voici ce que l'on écrirait dans la base de données, pour la page « À propos » :

%APP_NAME%	au lieu de	CCCVs-Transfert
%APP_VERSION%	au lieu de	1.0.0

La méthode « ***parmConverter*** » se chargera d'aller lire la valeur reçue par la base de données et remplacera grâce à une expression régulière toutes les valeurs contenus entre les '%' pour les remplacer par le nom des paramètres de la table « ***trans_app_param*** ».

ATTENTION !

A travers la base de données, il faut bien faire la différence entre les paramètres en minuscules (`%author%`) et ceux en majuscules (`%APP_VERSION%`).

Les paramètres notés en minuscules ne sont pas convertis par rapport aux paramètres de la table « trans_app_param ». Ce sont des paramètres qui sont remplacés dans le code par une valeur variable, par exemple l'adresse e-mail qu'entre un utilisateur.

L'avantage de cette méthode est de pouvoir re-factoriser les contenus. De cette manière, l'application offrira à ses futurs administrateurs la plus grande souplesse possible. Le but étant de n'avoir quasiment jamais à recompiler l'application pour effectuer des modifications. Tout, ou presque, à long terme sera intégré à la liste des paramètres de la base de données pour atteindre cet objectif d'édition simplifiée.

7. TRADUCTION DE L'APPLICATION

L'application est disponible en français et en allemand. La traduction est possible grâce à la table « `trans_translate` » qui contient tous les textes en français-allemand ainsi qu'une méthode nommée « `i` » (internationalize) de la class « `Global` ».

La raison d'une méthode au nom si court est simple : elle est abondamment utilisée à travers toute l'application et il convient de lui donner un nom le plus court possible pour éviter d'avoir un code énorme qui exécute des actions simples. L'utilisation et le nommage d'une telle méthode m'est inspirée d'une expérience de développement sur le logiciel WinBiz qui est actuellement traduit en 4 langues et qui utilise exactement le même principe.

Pour une traduction véritable des textes dans les différentes langues, deux options s'offrent à moi :

- Utiliser une méthode du type `Global.i("Voici mon texte") ;` qui irait chercher directement la ligne correspondant à ce texte dans la base de données et sélectionnerait le champs de la langue choisie par l'utilisateur.

  Avantage : Meilleure lisibilité et compréhension du code
  Inconvénient : Si le texte est modifié dans la base de données, l'application ne l'affichera plus car il ne trouvera plus de texte correspondant, il faudra donc apporter les mêmes modifications dans le code et recompiler le programme. Ce n'est pas ce que nous voulons.
- Utiliser une méthode du type `Global.i("CAPTION_DOWNLOAD") ;` où « CAPTION_DOWNLOAD » est le mot clé du texte à afficher. Ce mot clé, qui restera fixe, permettra d'aller chercher le texte désiré.

  C'est cette dernière option que j'ai retenu pour traduire cette application.

La méthode `i` dont j'ai parlé fonctionne d'une manière simple. Elle vérifie tout d'abord si le HashMap qui va contenir les textes est vide. Si tel est le cas, elle va le remplir en parcourant toute la table `trans_translate`, sinon, elle ne va qu'afficher la valeur demandée. Ainsi, chaque fois que cette méthode sera appelée, elle n'aura pas à exécuter de requêtes SQL, elle ne le fera que la toute première fois.

7.1. RECONNAISSANCE DE LA LANGUE DU NAVIGATEUR

Lors de l'inscription des utilisateurs, plusieurs informations les concernant sont mémorisées dans la base de données. Une d'entre elle est la langue du navigateur. Celle-ci sera considérée comme sa « langue préférée » et utilisée par défaut lorsque celui-ci se connectera à l'application.

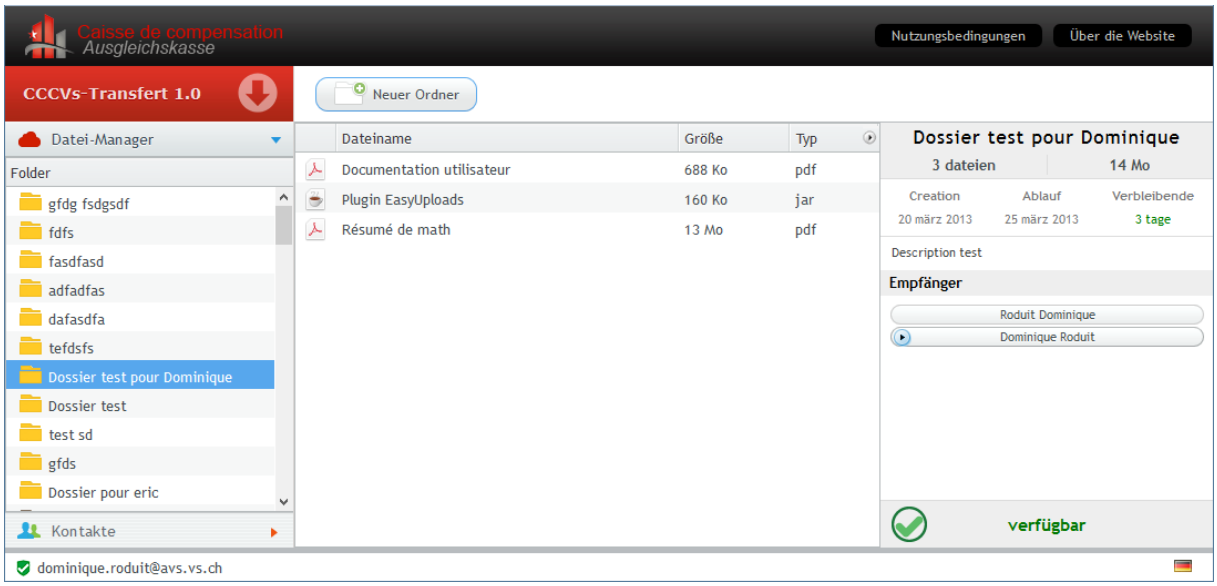
La langue du navigateur nous est fournie par un objet Vaadin, représenté sous forme de champ, accessible depuis la class « Global ». Pour récupérer la langue du navigateur, il suffit d'écrire : `Global.browser.getLocale().`

7.2. TRADUCTION COMMANDÉE

L'utilisateur peut décider de changer la langue d'affichage de l'application à tout moment en cliquant sur le drapeau situé dans le bord inférieur droit de l'interface utilisateur.

Sur le clic de ce drapeau, l'interface est entièrement réaffichée et l'objet HashMap contenant les textes entièrement rechargé. La session est également mise à jour.

Voici un aperçu de l'interface lorsqu'elle est traduite en allemand :

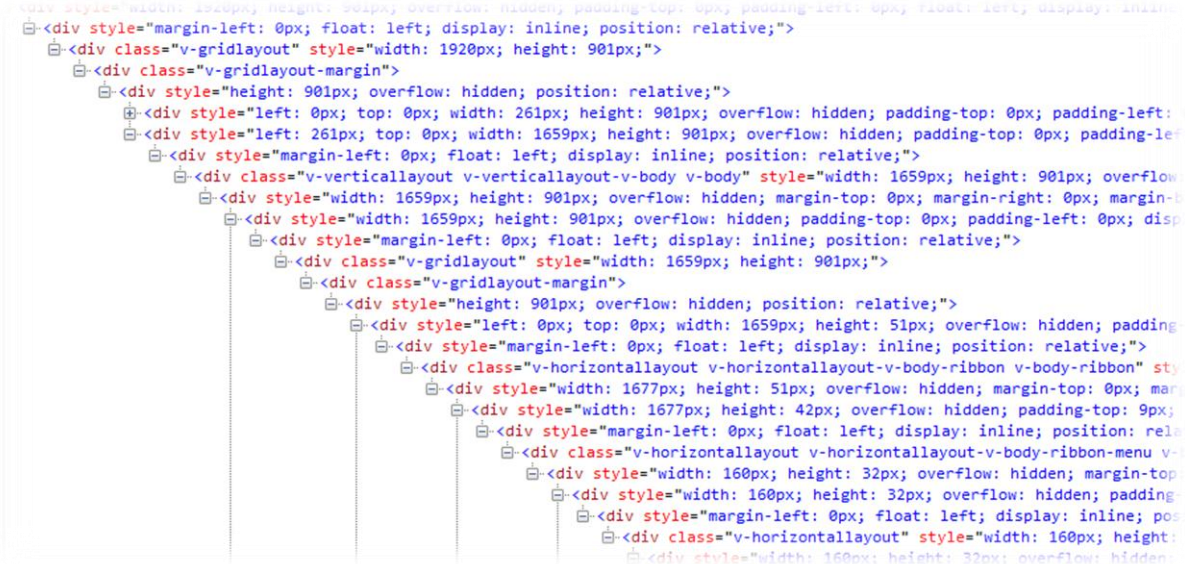


Pour la phase de développement, l'application a été traduite par moi-même à l'aide de Google Translator, les traductions ne sont donc pas totalement fiables.

8. CRÉATION DU DESIGN

Avec Vaadin, il n'y a pas de problèmes de compatibilité entre les navigateurs. Une fois le concept assimilé, la création du design est relativement facile et rapide.

En revanche, le code est long. Le moindre attribut d'un layout doit être renseigné et la librairie génère énormément de DIV – souvent inutiles – pour un seul layout, voire même pour un seul Label, ce qui ralenti drastiquement le chargement des pages et rend plus difficile le débogage. Cette capture d'un code généré par Vaadin, illustre parfaitement mes propos.



Voici quelques concepts Vaadin fondamentaux à comprendre pour la création d'un design complexe.

HorizontalLayout

Ce n'est pas comme on pourrait le penser, un bloc orienté horizontalement. C'est un bloc qui va positionner tous les composants contenus horizontalement. Lorsqu'on passe du monde de la programmation web au Vaadin, ce concept ne paraît pas logique au premier abord. C'est pour cette raison que je tiens à le mentionner. Dans un formulaire avec des légendes, où il est utile d'aligner les éléments horizontalement, il est préférable d'utiliser ce layout.

VerticalLayout

Même principe que pour le layout Horizontal, celui-ci alignera les éléments qu'il contient Verticalement.

GridLayout

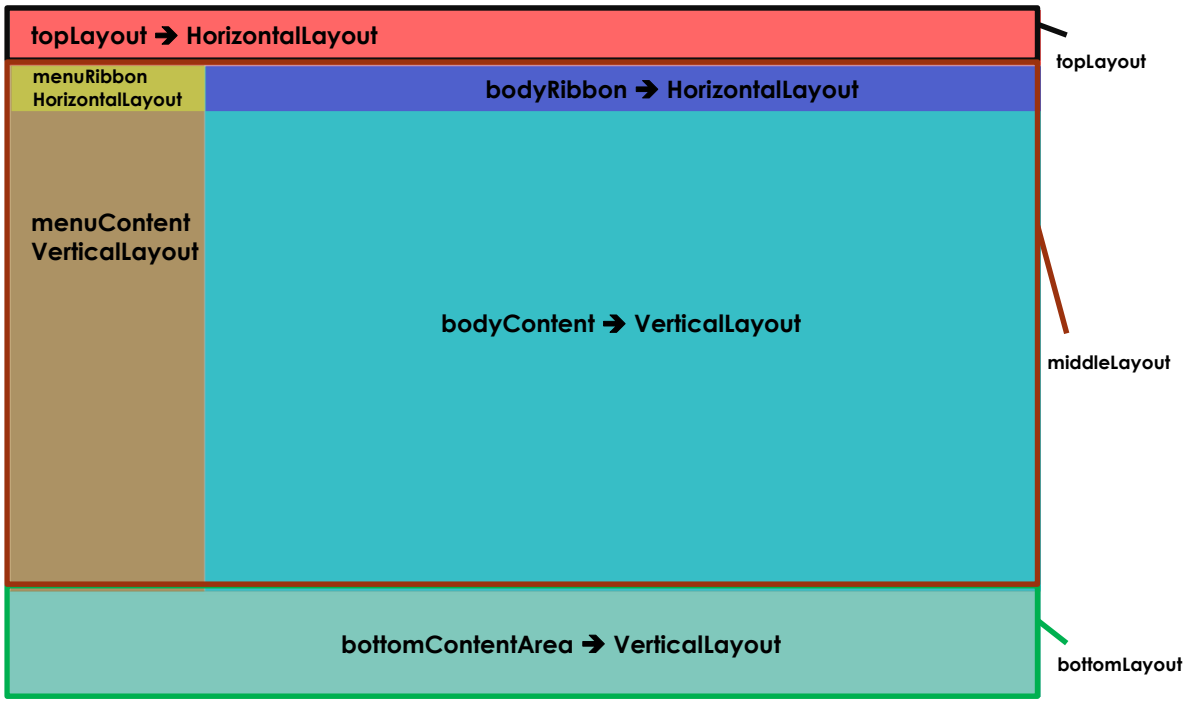
Le GridLayout n'est pas, comme on pourrait le penser, un tableau généré en HTML. C'est une série de DIV formatées pour placer les éléments en lignes et en colonnes. Ce composant est également utile de par ses attributs « `setColumnExpandRatio` » et « `setRowExpandRatio` », lorsqu'un layout doit prendre la totalité de la largeur ou de la hauteur de l'écran.

AbsoluteLayout

C'est un layout qui « flotte ». En CSS, on pourrait le comparer à une DIV en position absolue. Par défaut, il se place à l'origine du composant parent, à la coordonnée (0;0).

8.1. SCHÉMA DES LAYOUTS

Le schéma ci-après représente le positionnement des différents layout ainsi que leurs noms et types.



9. AUTHENTIFICATION DES UTILISATEURS

Avant l'étude initiale, l'application n'était pas censée contenir une page d'authentification. Les utilisateurs devaient être identifiés simplement par leur adresse e-mail. Cette méthode ne me paraissait pas très satisfaisante pour une application dont un des objectifs est la sécurité. De plus, elle ne permettait pas de vérifier l'authenticité d'une adresse e-mail. Il m'a été demandé de ne pas envoyer de ticket de téléversement, méthode qui a normalement pour objectif de vérifier l'adresse des différents partis impliqués dans un transfert. On aurait donc pu sans peine utiliser l'adresse e-mail d'une tierce personne à des fins nuisibles. Le premier jour du projet, avant le lancement de la conception, Monsieur Lorétan et moi-même en avons donc conclut qu'il était important de doter notre système d'une page de connexion. Ceci implique également une inscription de la part des utilisateurs.

Le but de ce chapitre est de vous décrire le comportement de la page de connexion/inscription ainsi que les méthodes utilisées.

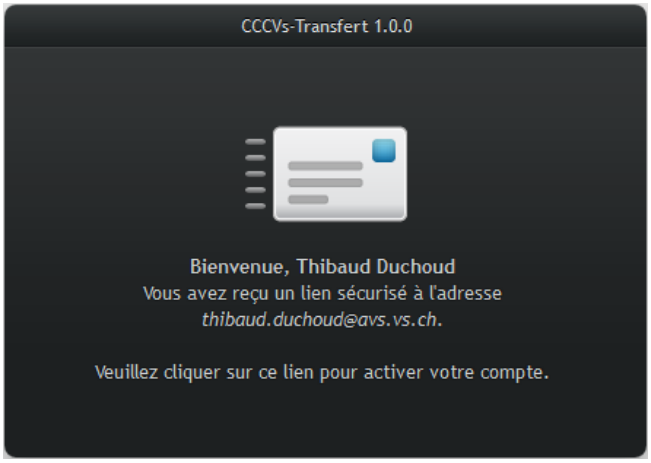
9.1. INSCRIPTION

La procédure « standard » sans faute d'une inscription est la suivante :

L'utilisateur n'est pas inscrit. Il saisit son adresse e-mail et choisi son mot de passe. Avant même d'entrer dans le champ de saisie du mot de passe, le bouton de validation

apparaît avec l'intitulé « Inscription ». Il suffit de cliquer sur ce bouton ou de valider le formulaire par la touche ENTER pour soumettre les informations, qui seront contrôlées puis validées ou refusées.

Dans le cas d'une inscription « sans faute », l'application indique qu'un mail a été envoyé à l'adresse spécifiée et qu'il convient de valider le lien contenu dans ce message pour être autorisé à poursuivre vers l'étape de connexion.



9.1.1. CRYPTAGE DU MOT DE PASSE

Lors de l'inscription de l'utilisateur, son mot de passe est crypté à l'aide de la fonction `getHashValue` de la class `Utilities.SHA256`.

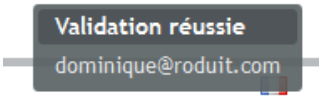
Le mot de passe n'est pas conservé en clair dans la base de données. Seule la valeur hachée est stockée. Le mot de passe est donc modifiable par l'administrateur du système mais irrécupérable si l'utilisateur ne s'en souviens plus.

Lors des connexions, le mot de passe saisi est haché en SHA256. Cette valeur est comparée au mot de passe haché de la base de données. Si la valeur n'est pas la même, le mot de passe n'est pas bon et un message d'erreur est affiché à l'écran :

Pour de plus amples informations concernant les méthodes de cryptage utilisées, consultez le chapitre 13. en page 45.

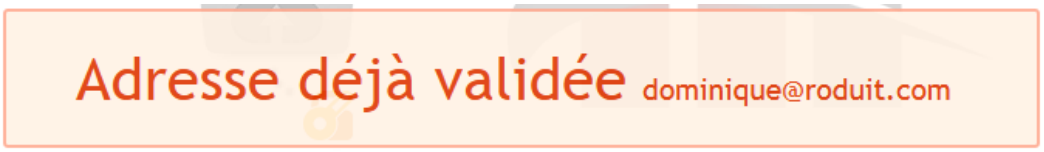
9.2. VALIDATION

Dès que les utilisateurs s'inscrivent, un mail leur est envoyé. Celui-ci contient un lien de validation crypté. Il leur suffit de suivre ce lien pour que leur compte soit validé. Lors de la validation, l'utilisateur sera automatiquement connecté à l'interface.



Le chapitre 13.2. détaille la manière de procéder pour intercepter, analyser et sortir les informations de la base de données correspondantes aux paramètres reçus dans l'URL.

Lorsque l'utilisateur a validé son compte et qu'il tente de le re-valider en cliquant à nouveau sur le lien reçu, le message d'erreur suivant est affiché :



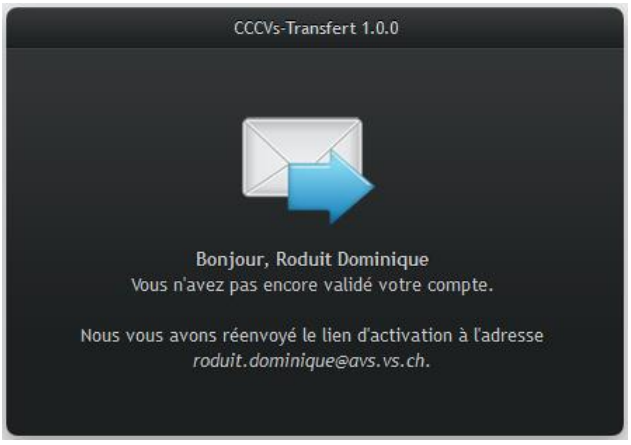
Si un individu mal intentionné tenterait de modifier le paramètre de l'URL, le système de comparaison des valeurs cryptées détecte qu'aucun compte ne correspond à la valeur du paramètre réceptionné et celui-ci est ignoré.

9.3. CONNEXION

Cette section décrit les différents cas qui peuvent survenir lorsqu'une connexion est sollicitée par un utilisateur.

9.3.1. COMPTE NON-VALIDÉ

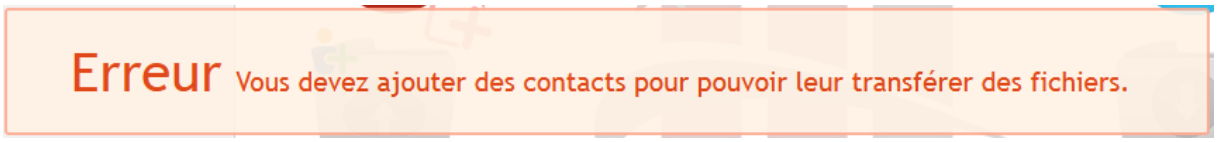
Si l'utilisateur s'est inscrit, n'a pas encore validé son compte et tente de se connecter, la fenêtre suivante lui est affichée et le mail de validation renvoyé dans sa boîte de réception.



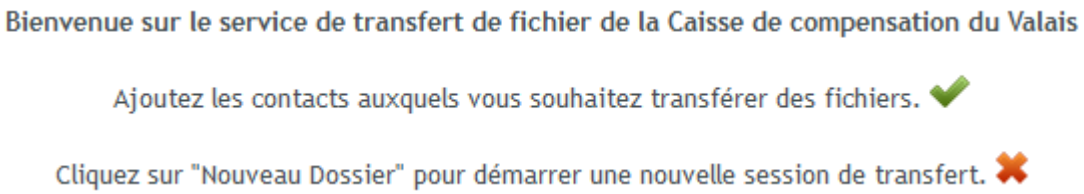
9.3.2. PREMIÈRE CONNEXION

Lors de la première connexion, l'utilisateur est guidé à travers l'application. La page d'accueil lui indique les opérations à effectuer pour pouvoir transférer des fichiers correctement. Voici comment se présente la page sur laquelle l'utilisateur arrive lors de sa toute première connexion :

S'il tente de créer un nouveau dossier avant d'avoir ajouté un contact, il est directement redirigé vers la page d'ajout des contacts et un message l'averti qu'il doit ajouter au minimum un contact pour pouvoir transférer des fichiers (sinon il n'existe aucun destinataire).



Lorsque l'utilisateur a ajouté un contact, la condition pour pouvoir créer un nouveau dossier est remplie :



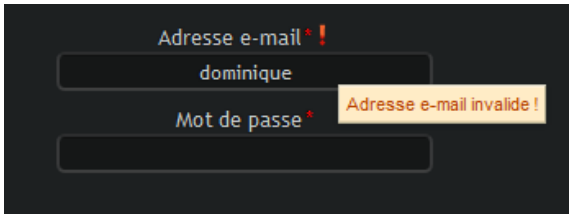
9.4. AUTRES CAS D'ERREURS

Dans le cas où une saisie est incorrecte (vide, dans un format inadapté,...) un message d'erreur informe de la modification à apporter.

Exemples :

Mot de passe trop court

Adresse e-mail dans un mauvais format



9.5. JOURNALISATION DES OPÉRATIONS

Voici la liste des actions journalisées dans la table « *trans_journal_connections* » ainsi que le nom et la description de l'action tels qu'ils sont enregistrés dans la base de données.

L'adresse e-mail du compte concerné est également enregistrée à chaque fois.

Aucune information n'est saisie dans le formulaire

Nom de l'action con_invalid_infos

Message enregistré xxx a tenté de se connecter en entrant des informations invalides.

L'adresse e-mail est saisie dans un format invalide

Nom de l'action con_invalid_mail

Message enregistré xxx a tenté de se connecter en entrant une adresse e-mail invalide.

L'adresse e-mail saisie n'existe pas dans la base de données

Nom de l'action con_new_user

Message enregistré xxx n'existe pas encore dans la base de données, on lui envoie un mail avec un lien sécurisé.

L'adresse e-mail saisie est connue mais n'est pas encore validée

Nom de l'action con_no_validate

Message enregistré xxx a entré un bon mot de passe mais n'a pas encore validé son adresse, on lui renvoie un mail avec le lien sécurisé.

L'adresse e-mail est connue et validée, mais le mot de passe est erroné

Nom de l'action	con_wrong_pass
Message enregistré	xxx existe mais a entré un mauvais mot de passe.

L'adresse e-mail saisie est connue, validée et le mot de passe est bon

Nom de l'action	con_success
Message enregistré	xxx a accès, il a entré un bon mot de passe et son adresse est validée, on le connecte.

Une exception SQL est déclenchée

Nom de l'action	con_error_exception
Message enregistré	xxx a déclenché une exception : erreur de connexion à la base de données. Pass : ********.

Validation réussie

Nom de l'action	validation_success
Message enregistré	Validation réussie

Validation déjà effectuée

Nom de l'action	validation_recurrent
Message enregistré	Relancement du lien alors que l'adresse est déjà validée.

Validation d'un compte inexistant

Nom de l'action	validation_error
Message enregistré	Essai de validation d'une adresse qui n'existe pas

10. DIFFUSION DES MESSAGES ÉLECTRONIQUES

L'application CCCVs-Transfert doit correspondre avec les différents usagers du programme. Dans de nombreuses situations, il est nécessaire de transmettre d'importantes données. Celles-ci doivent être consultables par le destinataire dans les plus brefs délais. L'envoi de messages électroniques répond à ce besoin. Voici les différentes démarches qui engendrent l'expédition d'un courrier électronique :

1 Validation des comptes

Lors de l'inscription, l'utilisateur reçoit automatiquement un message électronique. Celui-ci contient un lien sécurisé qui permet de valider le compte. Cette action est requise pour éviter d'accueillir n'importe quel robot sur la plateforme de transfert. L'utilisateur ne peut valider son compte qu'une seule fois. Si ce dernier tente de relancer le lien de validation à plusieurs reprises, un message lui indique que son compte est déjà validé.

Aperçu d'un message reçu :


Validation de votre compte
Boîte de réception x
CCCVs-Transfert <support@avs.vs.ch>
à moi
CCCVs-Transfert
Bonjour,
Voici le lien pour valider votre inscription sur notre interface de transfert de fichiers.
Valider mon inscription
Ce message vous a été envoyé automatiquement, merci de ne pas y répondre.
Caisse de compensation du Valais
Ausgleichskasse Wallis

2 Accès à un dossier de téléchargement

Lorsque l'utilisateur de l'application distribue son dossier à un ou plusieurs destinataires, un e-mail est envoyé à chaque entité mentionnée. Chaque message contient un lien avec des paramètres cryptés. Le paramètre relatif au dossier est le même pour tous les utilisateurs. Mais deux autres chaînes sont également stockées dans l'URL pour permettre la journalisation des accès et téléchargements. Pour des informations plus détaillées à propos du cryptage et des accès aux dossiers, veuillez consulter le chapitre 14.1. en page 48.

Aperçu d'un message reçu :


Lien de téléchargement
Boîte de réception x
CCCVs-Transfert <support@avs.vs.ch>
à moi
CCCVs-Transfert
Bonjour,
dominique.roduit@avs.vs.ch vous a partagé des fichiers depuis le service de transfert de la Caisse de compensation.
Description test
Ce dossier est disponible 5 jours à partir d'aujourd'hui.
Date d'expiration : 25.03.2013.
Voici le lien pour accéder au dossier qu'il vous a destiné.
Télécharger les fichiers
Ce message vous a été envoyé automatiquement, merci de ne pas y répondre.
Caisse de compensation du Valais
Ausgleichskasse Wallis

3 Rappel avant la suppression d'un dossier

Un jour avant la suppression d'un dossier (seulement ceux ayant une durée de vie initiale minimum à deux jours), un message électronique est envoyé à son auteur ainsi qu'à ses destinataires. Ce message est un rappel avant la suppression définitive des fichiers et l'archivage du dossier. Le mail envoyé à l'auteur contient un historique résumé et détaillé des actions effectuées par les utilisateurs sur le dossier. Le mail envoyé aux destinataires contient également un historique des actions effectuées uniquement par eux-mêmes.

Aperçu d'un message envoyé à l'auteur d'un dossier :


Rappel avant l'expiration de votre dossier

Boîte de réception x

CCCVs-Transfert <support@avs.vs.ch>
à moi

CCCVs-Transfert

Bonjour,

nous vous rappelons que votre dossier 'Dossier médical de Monsieur Olivier' expirera le 22.03.2013 à 08:00.
A partir de cette date, les fichiers seront supprimés et votre dossier ne sera plus accessible par les destinataires.

Voici l'historique des téléchargements pour chaque destinataire :

Historique résumé
Roduit Dominique
Aucun accès ni téléchargement

Historique détaillé
Roduit Dominique
Aucun accès ni téléchargement

Aperçu d'un message envoyé aux destinataires :


Rappel avant la suppression du dossier

Boîte de réception x

CCCVs-Transfert <support@avs.vs.ch>
à moi

CCCVs-Transfert

Bonjour,

nous vous rappelons que le dossier 'Dossier médical de Monsieur Olivier', partagé par dominique.roduit@avs.vs.ch expirera le 22.03.2013 à 08:00.
A partir de cette date, les fichiers seront supprimés et le dossier ne sera plus accessible.

Voici l'historique des évènements vous concernant enregistrés pour ce dossier :

- Historique vide -

Ce message vous a été envoyé automatiquement, merci de ne pas y répondre.

Caisse de compensation du Valais
Ausgleichskasse Wallis

4 Suppression d'un dossier

Avant l'archivage d'un dossier et des fichiers qu'il contient, un message est envoyé à chaque entité concernée (auteur et destinataires) pour avertir chacun d'eux de la suppression prochaine des fichiers du dossier. A partir de cette date, le dossier ne sera plus accessible par les destinataires. Cependant, le dossier restera visible par son créateur à titre indicatif uniquement. Il peut supprimer les dossiers archivés à tout moment puisque ceux-ci ne sont de toute façon plus accessibles par les destinataires précédemment agréés.

Aperçu d'un message envoyé à l'auteur d'un dossier :

Expiration de votre dossier Boîte de réception x

CCCVs-Transfert «support@avs.vs.ch»
à moi

CCCVs-Transfert

Bonjour,

votre dossier 'Dossier médical de monsieur Olivier' a expiré aujourd'hui (17.03.2013), à 08:00.
Les fichiers contenus ont été supprimés de notre serveur.
Ce dossier n'est donc plus visible pour les contacts auxquels vous aviez autorisé l'accès.

Voici l'historique des actions pour chacun des contacts :

Historique résumé

Roduit Dominique		
1	Archive du dossier	8x
5	putty_20130314094420.exe	1x
6	dominiqueroduit_20130314094214.crt	1x
7	dominiqueroduit_20130314100928.crt	2x
19	folder_caeeas_20130314.zip	1x
Dominique Roduit		
1	Consultation	1x

Historique détaillé

Roduit Dominique			
1	Consultation	Consultation du dossier	14.03.2013 18:02
2	Consultation	Consultation du dossier	14.03.2013 18:00

Aperçu d'un message envoyé aux destinataires :

Expiration du dossier partagé Boîte de réception x

CCCVs-Transfert «suppport@avs.vs.ch»
à moi

CCCVs-Transfert

Bonjour Roduit Dominique,

le dossier 'Dossier médical de monsieur Olivier' que dominique.roduit@avs.vs.ch vous avait partagé a expiré aujourd'hui (17.03.2013), à 08:00.
Les fichiers de ce dossier ont été supprimés de notre serveur. Par conséquent, vous n'y avez plus accès.

Voici l'historique des évènements vous concernant enregistrés pour ce dossier :

Consultation : 42x

	Fichier téléchargé	Date
1	Archive du dossier	14.03.2013 17:53
2	easyuploads1.0.0_20130314104205.jar	14.03.2013 17:52
3	Archive du dossier	14.03.2013 17:52
4	easyuploads1.0.0_20130314100928.jar	14.03.2013 17:46
5	Archive du dossier	14.03.2013 17:45
6	window_20130314101418.css	14.03.2013 17:44
7	proces-verbal_20130314094214.pdf	14.03.2013 17:14
8	dominiqueroduit_20130314094214.crt	14.03.2013 17:12
9	putty_20130314094214.exe	14.03.2013 17:11
10	putty_20130314094214.exe	14.03.2013 17:11
11	Archive du dossier	14.03.2013 15:26

10.1. ENVOI DE MESSAGES ÉLECTRONIQUES

10.1.1. PRÉREQUIS

Pour envoyer des messages électroniques en Java, il est important d'intégrer la librairie `JavaMail` qui peut être téléchargée sur internet.

Il faut également un serveur SMTP capable d'envoyer des messages à n'importe quelle adresse e-mail sur internet. Pour tester l'application, un petit logiciel gratuit – `MiniRelay` – a été utilisé à titre de serveur SMTP (uniquement sur le poste de développement). Il ne permet pas l'envoi de beaucoup de messages en parallèle, mais il est largement suffisant pour le besoin que nous avons.

Sur le serveur de production, il a également fallu installer un serveur SMTP puisque c'est sur ce serveur que l'application sera localisée lorsqu'elle sera publiée. Le logiciel PostFix a été privilégié pour sa simplicité d'utilisation et sa gratuité. Il a été installé avant l'exécution du projet lors de la mise en place de l'infrastructure système/réseau.

PostFix doit être configuré pour envoyer les mails vers un relai SMTP qui lui-même s'occupera de les distribuer aux destinataires. La configuration de PostFix s'effectue dans le fichier `/etc/postfix/main.cf`.

10.1.2. PROCÉDURE

L'envoi des e-mails se fait depuis une class Java implémentant la librairie JavaMail. Cette class se trouve dans le package `toolbox.Mailer`. La méthode principale de cette class est la suivante :

```
send(String from, String from_name, String to, String subject, String content)
```

Elle permet l'envoi de messages électroniques d'un auteur que l'on peut nommer par le paramètre « `name` » à un ou plusieurs destinataires spécifiés via l'attribut « `to` ». Les différentes adresses spécifiées dans cet attribut doivent être séparés par des virgules. Pour plus d'informations à propos de l'utilisation de la class `Mailer`, veuillez vous référer à la JavaDoc du package `Utilities`.

Cette méthode contient également les différents paramètres de configuration – tel le port d'exécution du serveur SMTP - qui permettent l'envoi d'un mail.

10.1.3. COMPOSITION D'UN MESSAGE TYPE

Les paramètres enregistrés dans la base de données peuvent être utilisés pour composer les mails envoyés. Deux paramètres sont importants :

- MAIL_HEADER_CONTENT Contenu et mise en page du header des messages
- MAIL_FOOTER_CONTENT Contenu du bas de page des messages

Ces deux paramètres doivent être utilisés dans tous les mails envoyés. La mise en page est ainsi utilisée uniformément dans tous les mails envoyés.

Presque tous les mails envoyés par l'application contiennent des contenus cryptés. Pour comprendre le procédé de cryptage des contenus, consultez la section *13. Cryptage des paramètres de l'URL (page 45)*.

10.1.4. Problèmes rencontrés

10.1.4.1. DMZ

Le fait que la DMZ ne soit pas configurée par l'état du Valais au moment où ces lignes sont écrites est quelque peu bloquant pour la réalisation finale du travail car la diffusion de messages sur le réseau externe (internet) n'est pas possible. Pour tous les tests effectués jusqu'à ce jour, ce ne sont donc que les adresses internes à la Caisse de compensation qui ont été utilisées puisque ce sont les seuls à recevoir les mails du serveur SMTP.

10.1.4.2. Port d'exécution

Il faut être attentif au port utilisé par le serveur SMTP pour la transmission des messages électroniques car le port 25 pose quelques problèmes. Laissant sortir les messages de n'importe quelle adresse e-mail sans vérification d'authentification, la réception des messages provenant de ce port sont souvent bloqués par les FAI pour éviter le Spam. C'est pourquoi il est préférable d'utiliser le port 587 qui demande une authentification et qui est la plupart du temps autorisé.

10.1.4.3. Relai SMTP

Lorsque la DMZ fut configurée par l'état du Valais (le 14 Mars 2013), j'ai effectué plusieurs tests, pensant que le problème d'envoi des mails vers les adresses extérieures serait résolu. Malheureusement, ce ne fut pas le cas. Je ne savais plus sur quelle voie me lancer pour résoudre ce problème. J'ai contacté la personne qui a effectué la configuration de la DMZ à l'état du Valais et lui ait exposé mon problème. Celle-ci m'a confirmé que le problème ne venait pas de la DMZ. Il m'a informé que je devais utiliser le relai SMTP « mail.vs.ch » et que je devais demander une autorisation de sortie au helpdesk de l'état du Valais. J'ai soumis la demande au helpdesk le 18 mars 2013.

10.1.4.4. Encodage des caractères

Les mails envoient des textes de la base de données. La base de données a son propre encodage. Lorsqu'on envoie un mail il faut donc utiliser le même encodage, sinon, les caractères accentués ne seront pas affichés, ou affichés d'une manière spéciale.

Pour résoudre ce problème, il y a deux manipulations importantes à réaliser :

MySQL

Le fichier de configuration de MySQL sur le serveur se localise par défaut dans `/usr/my.cnf`. Sauf que si ce fichier est édité, rien ne se passe car pour que les modifications soient prises en compte, il faut le copier dans `/etc/my.cnf` ! La base de données sur le poste de développement utilise un encodage `utf-8`. Voici les lignes à ajouter au fichier de configuration pour que les accents soient affichés correctement.

```
[client]
default-character-set=utf-8
[mysql]
default-character-set=utf-8
[mysqld]
character-set-server=utf-8
init-connect='SET NAMES utf-8'
```

Il est important ensuite de redémarrer le serveur MySQL avec la commande `/etc/init.d/mysql restart.`

Code Java

Dans la classe `Utilities.Mailer`, la méthode chargée d'envoyer les mails (`send`), doit indiquer quel encodage est utilisé pour les caractères envoyés. Voici donc ce qu'il a fallu ajouter :

```
message.setContent(content, "text/html; charset=UTF-8");
```

11. NAVIGATION

11.1. THREADLOCAL

Sur un site web standard, lorsqu'on est connecté et qu'on recharge la page, nous ne sommes pas subitement déconnectés. Il me semblait logique que mon application suive le même principe (malgré qu'elle ne change jamais d'URL et n'a donc pas besoin d'être rechargée manuellement par l'utilisateur).

Avec Vaadin, il est possible de « simuler » l'effet d'une session grâce à un « ThreadLocal ».

Le ThreadLocal va conserver l'état de l'application. Il va enregistrer une sorte de « vue figée » des objets instanciés, qu'il va redéployer lors du chargement pour éviter que l'affichage soit réinitialisé et que les utilisateurs soient chaque fois ramenés vers la page de connexion.

Néanmoins, si le paramètre `?restartApplication` est renseigné dans l'URL, l'application sera redémarrée quoi qu'il en soit car ce paramètre est conçu exprès pour forcer la réinitialisation de l'application.

Sans le paramètre ?restartApplication, l'utilisateur peut même rafraichir la page du navigateur avec la touche F5 durant un envoi de fichier, sans que celui-ci soit arrêté. Le ThreadLocal est donc très pratique dans ce cas de figure.

Je n'implémente qu'un seul ThreadLocal. Cet objet est instancié dans la classe principale de l'application (*CccvsTransfert*).

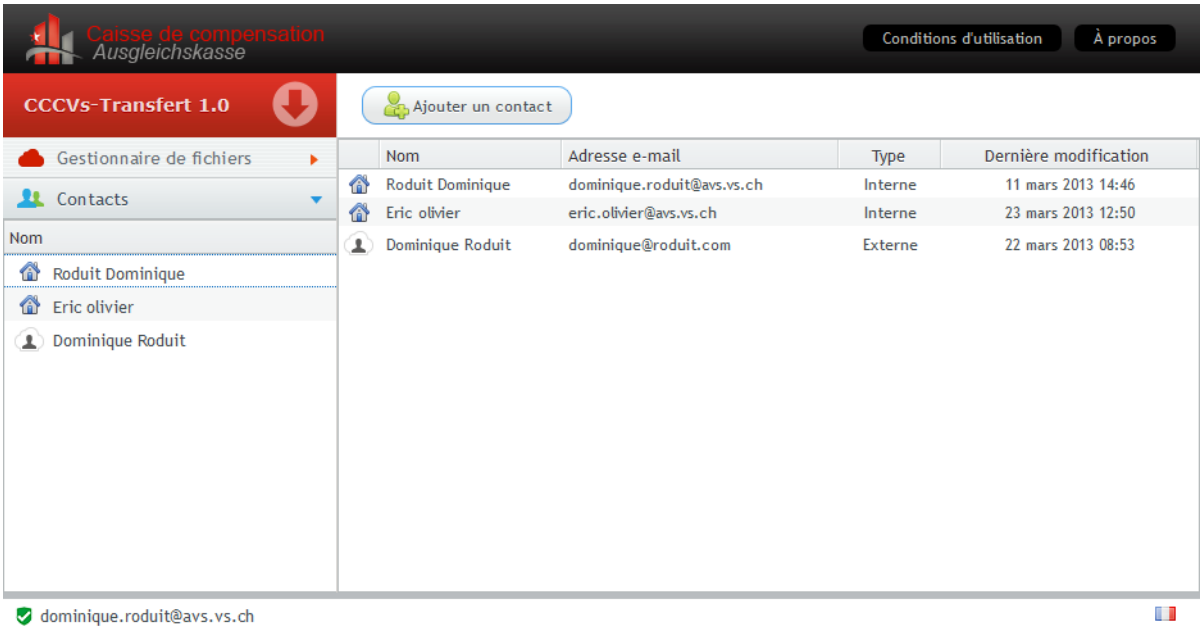
11.2. PAGE « CONTACTS »

La page de gestion des contacts est très basique car elle ne faisait pas partie du cahier des charges initialement. Elle est toutefois importante car c'est elle qui impose une première sécurité en n'autorisant pas les utilisateurs externes à ajouter des contacts autres que ceux possédant une adresse e-mail se terminant par « @avs.vs.ch ».

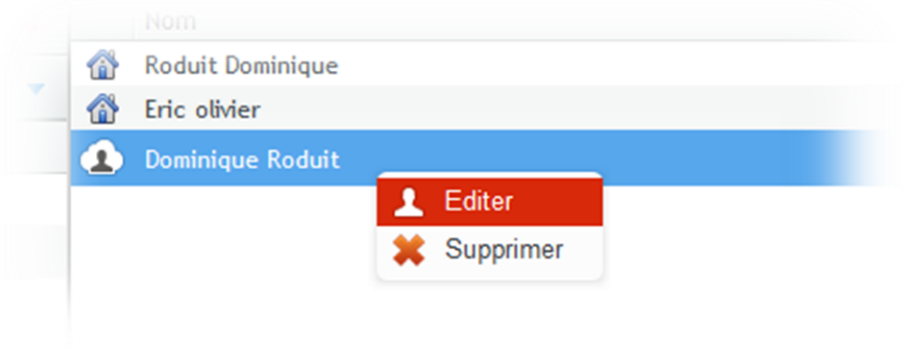
Si l'utilisateur est un interne, il pourra bien entendu ajouter les contacts qu'il souhaite sans restriction si ce n'est que le format de l'adresse e-mail doit être valide.

Voici un aperçu du formulaire d'ajout de contact :

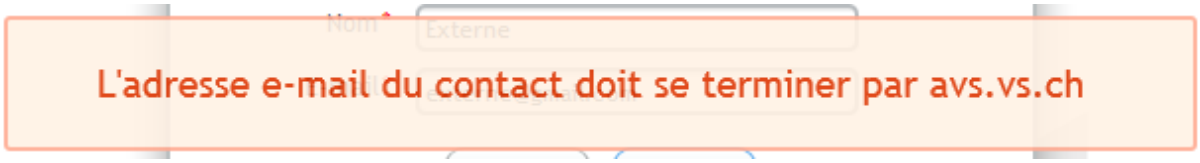
Lorsque le contact est ajouté, les listes des deux tableaux se mettent à jour, de même que lorsqu'un contact est édité ou supprimer depuis l'un ou l'autre de ces tableaux.



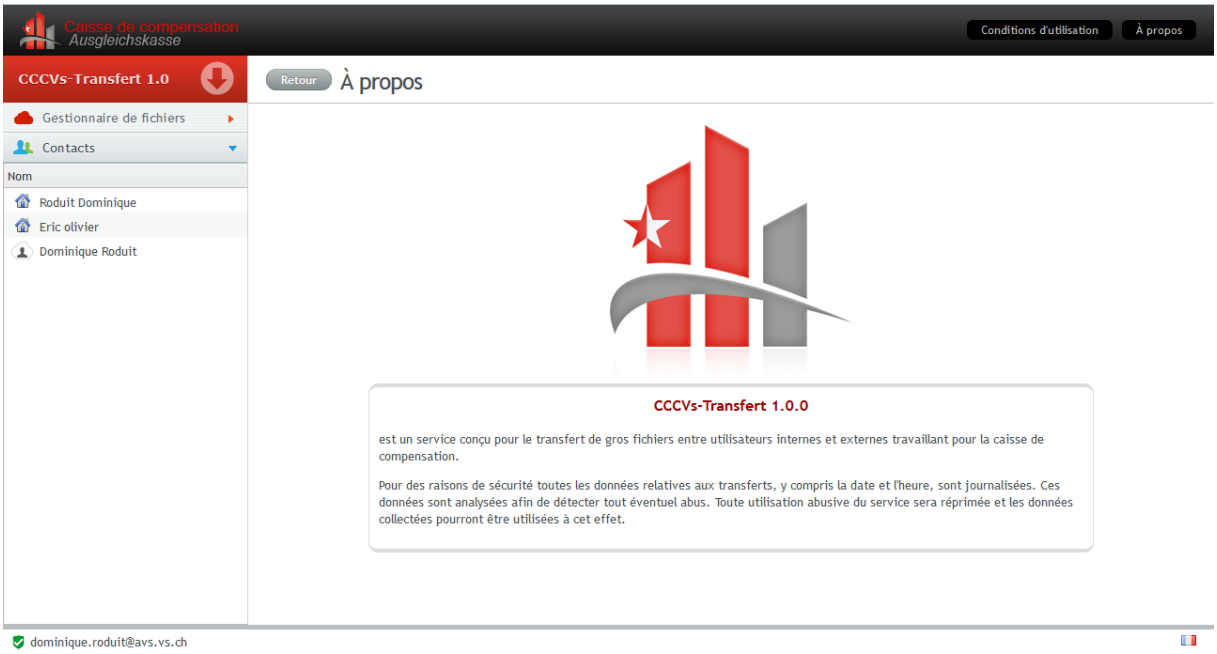
Voici les différentes actions que l'utilisateur peut entreprendre sur un contact en effectuant un clic droit sur la ligne en question.



Voici le message d'erreur affiché lorsqu'un utilisateur externe (ici, dominique@roduit.com), tente d'ajouter un contact externe (ici, externe@gmail.com) :



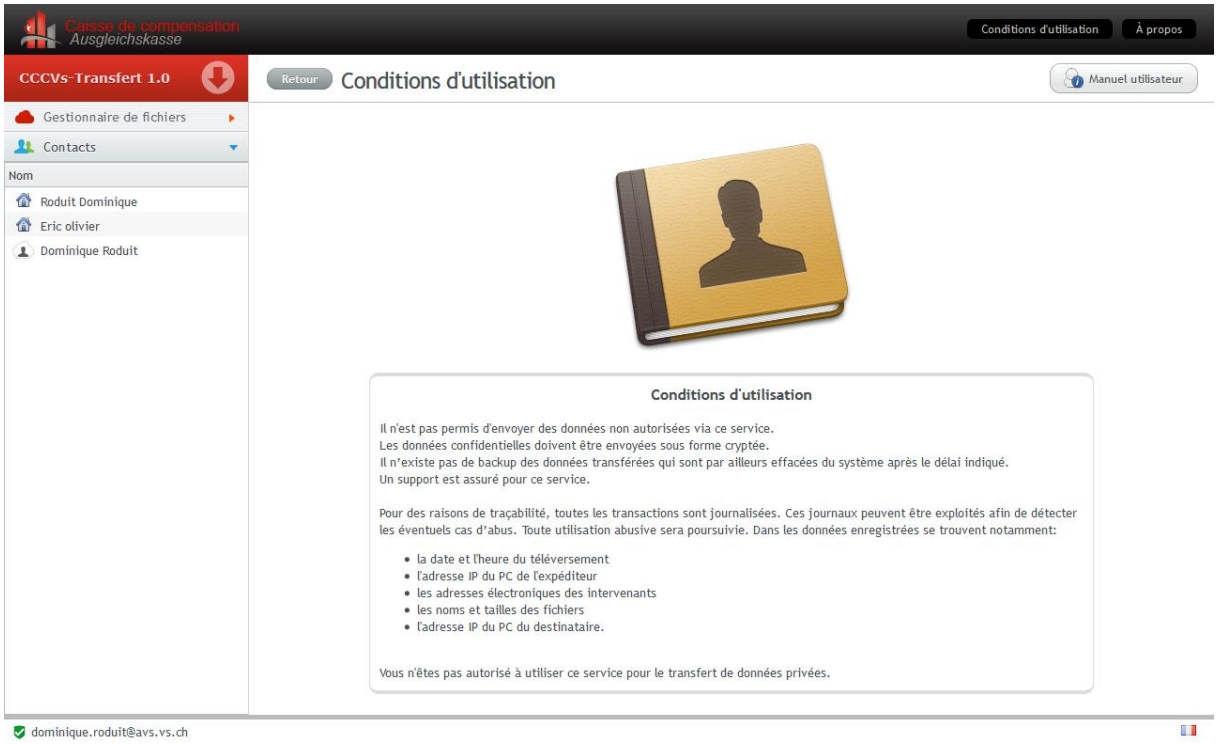
11.3. PAGE « À PROPOS »



Cette page informe l'utilisateur sur le propriétaire de l'application ainsi que sur la journalisation de ces données.

Le contenu de cette page ne fait qu'office d'exemple. Un texte officiel sera rédigé par la Caisse de compensation une fois l'application diffusée.

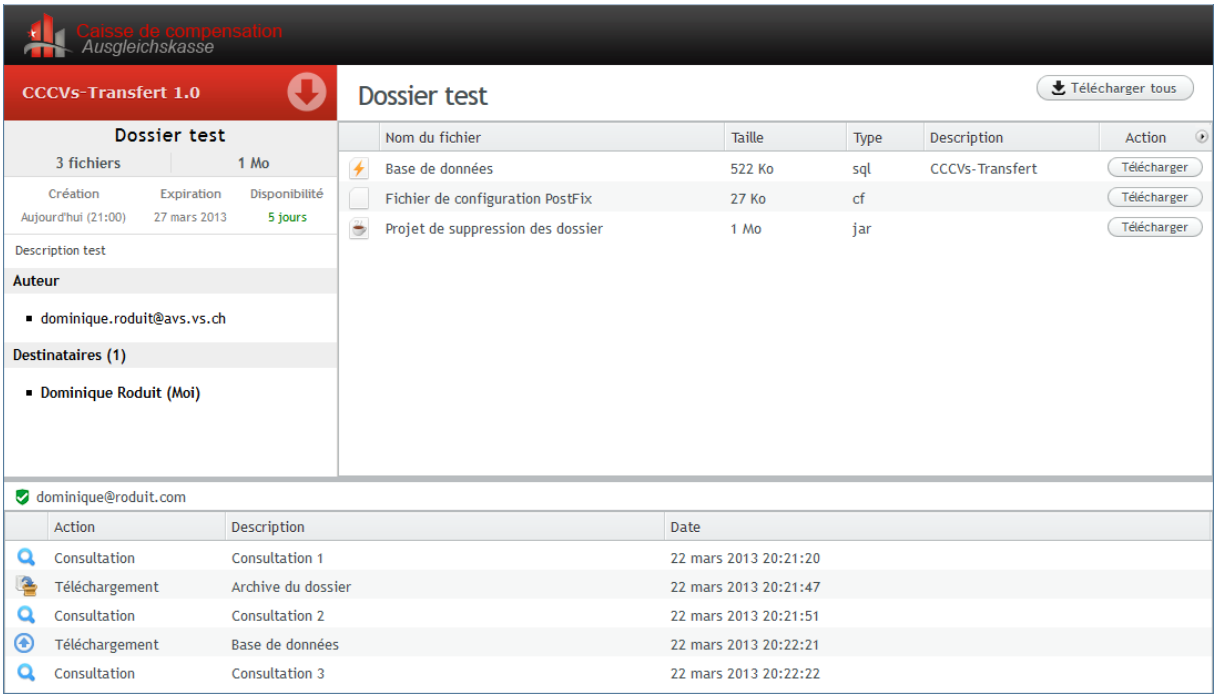
11.4. PAGE « CONDITIONS D'UTILISATION »



Comme son nom l'indique, cette page présente les conditions d'utilisation de l'application. Ces conditions ont été en parties copiées sur l'application WebTransfert et adaptées par moi-même. Le contenu de cette page ne fait qu'office d'exemple. Un texte officiel sera rédigé par la Caisse de compensation une fois l'application diffusée.

Le bouton « Manuel utilisateur » permet d'afficher le manuel d'utilisation dans une fenêtre flottante :

11.5. Page « Téléchargement »



Voici un aperçu de la page sur laquelle les destinataires invités à un dossier accèdent.

En haut de la page, dans le ruban, un bouton « Télécharger tous » permet de télécharger une archive au format ZIP de tous les fichiers contenus dans ce dossier.

Dans la partie latérale gauche on trouve les informations sur le dossier avec sa date de création, la date à laquelle le dossier va expirer ainsi que le nombre de jours (ou d'heures) restants avant l'expiration du dossier.

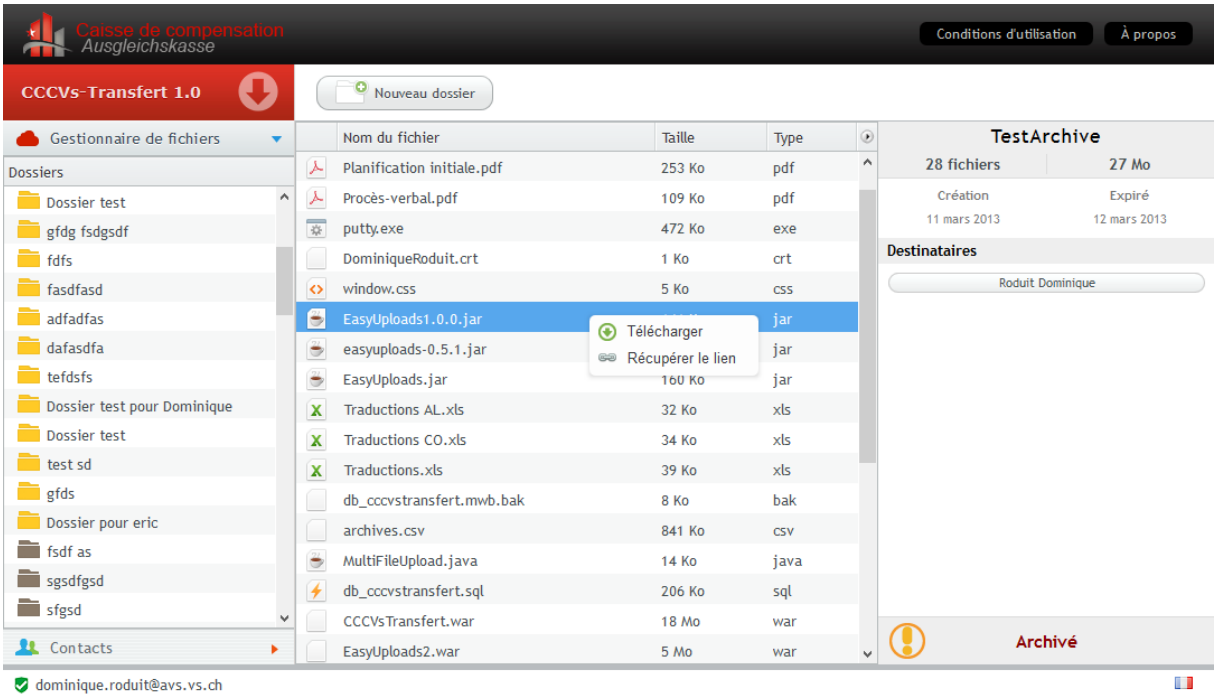
La partie du bas, affiche en temps réel les actions effectuées par le contact sur le dossier. Par exemple, dans ce cas-ci, le contact est arrivé sur la page (Consultation), il a cliqué sur « Télécharger tous » puis à télécharger uniquement le fichier « Résumé de math » à part.

Toutes ces actions sont rapportées à l'auteur du fichier, dans le mail qu'il reçoit avant la suppression du dossier.

Les fichiers peuvent également être téléchargés via un menu contextuel affiché sur le clic droit de la souris.

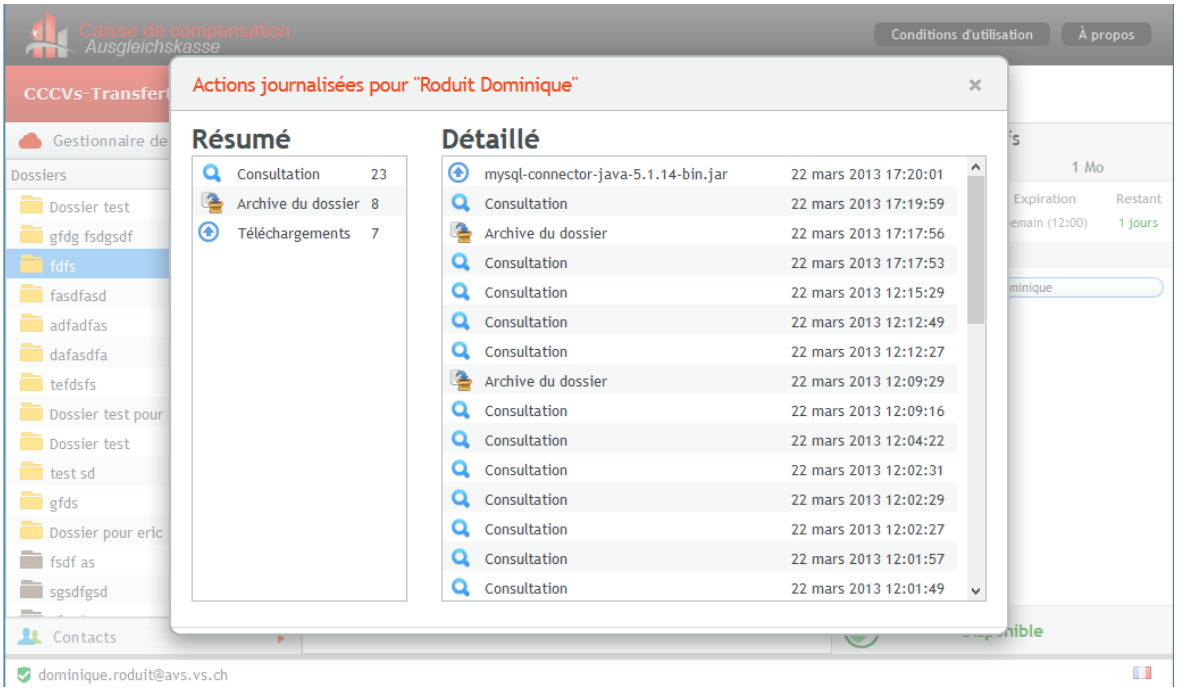
11.6. Page « Gestionnaire de fichiers »

Cette page inclus un menu qui répertorie tous les dossiers créés par l'utilisateur. Celui-ci peut consulter les informations de chaque dossier, les actions journalisées des destinataires qui y ont accédés. Il peut également télécharger ses propres fichiers, récupérer l'URL des liens vers les fichiers, supprimer les dossiers expirés, renvoyer les liens d'accès vers les dossiers, etc… Ce module est l'un des principaux et des plus élaborés après l'utilitaire de transfert de fichier. C'est cette page qui est affichée lors de la connexion de l'utilisateur (dès que celui-ci a créé au minimum un dossier).



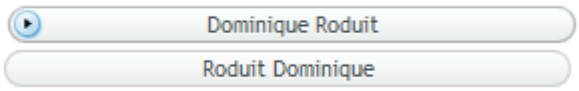
Les prochains chapitres présentent un peu plus en détail les fonctionnalités des modules qui sont accessibles depuis cette page.

11.7. AFFICHAGE DES ACTIONS JOURNALISÉES



Lorsque les destinataires accèdent au dossier, chaque action entreprise est journalisée dans la base de données. Ceci est utile pour une sécurité maximale mais permet également à l'auteur du transfert d'avoir un suivi détaillé du dossier. J'ai implémenté un module qui affiche un « historique » résumé et détaillé pour chacun des contacts qui ont accès à un dossier.

Dans l'interface utilisateur, ce module est accessible directement à l'intérieur des dossiers en cliquant sur un des destinataires :

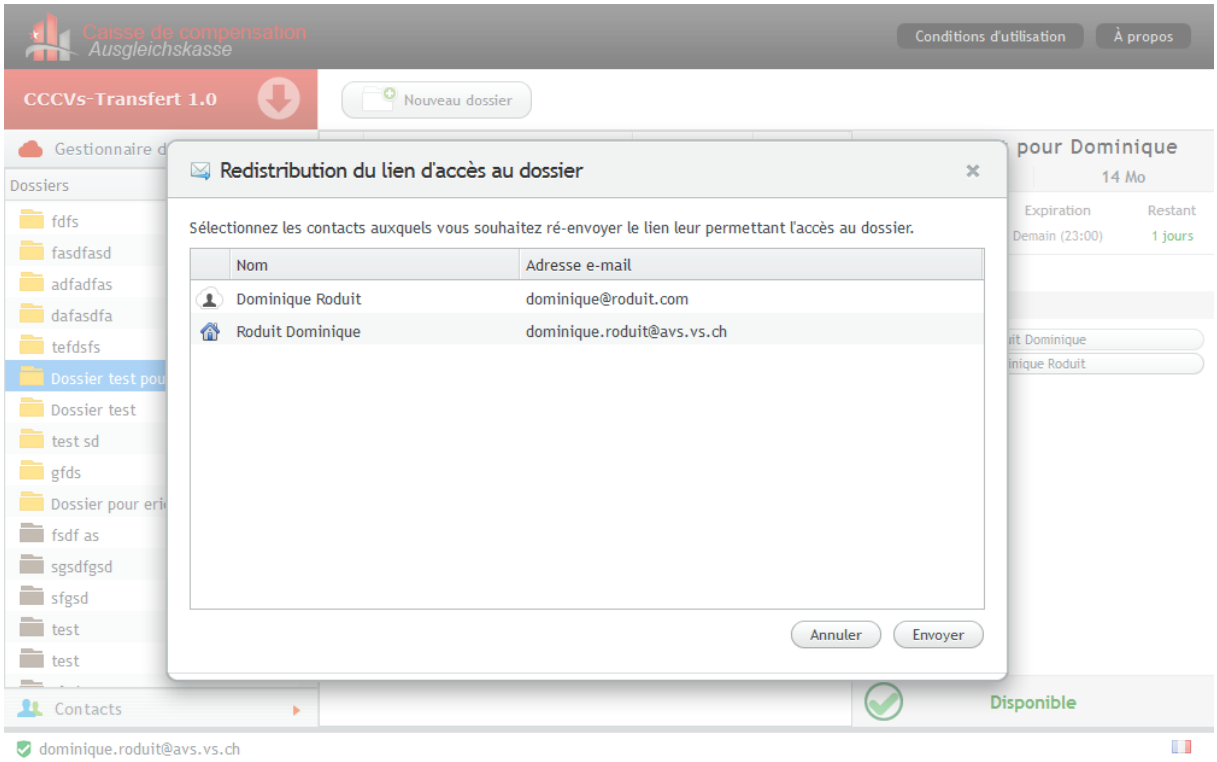


Dès que le contact aura consulté le dossier, l'icône bleu apparaitra à gauche de son nom. Le module n'est accessible qu'en présence de cette icône. Sur le clic d'un destinataire, si aucune action n'est enregistrée, le message suivant est affiché:

Les informations affichées dans ce module sont les mêmes que celles affichées au bas de la page de téléchargement d'un dossier. Ce sont également les mêmes informations qui sont envoyées par mail lors de la suppression d'un dossier.

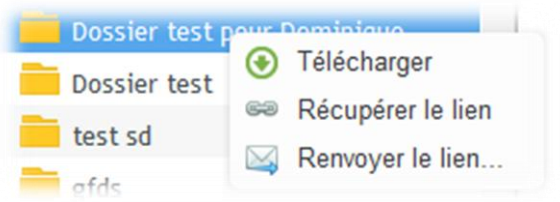
11.8. REDISTRIBUTION DES LIENS D'ACCÈS AUX DOSSIERS

Comment faire si un des contacts à qui on a envoyé un dossier se plaint de ne pas avoir reçu d'e-mail pour pouvoir y télécharger les fichiers ? Jusqu'ici, on était obligé de recréer un dossier avec les mêmes fichiers. Si les fichiers sont volumineux, cela peut vite s'avérer ennuyeux pour l'utilisateur. J'ai eu l'idée d'ajouter un petit module bien pratique pour pouvoir renvoyer les e-mails aux contacts désirés. Celui-ci se présente de la sorte :



Il suffit de sélectionner/désélectionner les contacts en cliquant sur eux dans le tableau puis de cliquer sur « Envoyer ». Si aucun contact n'est sélectionné, le message suivant est affiché :

Cette fenêtre est accessible directement depuis le menu contextuel affiché au clic droit sur un dossier.

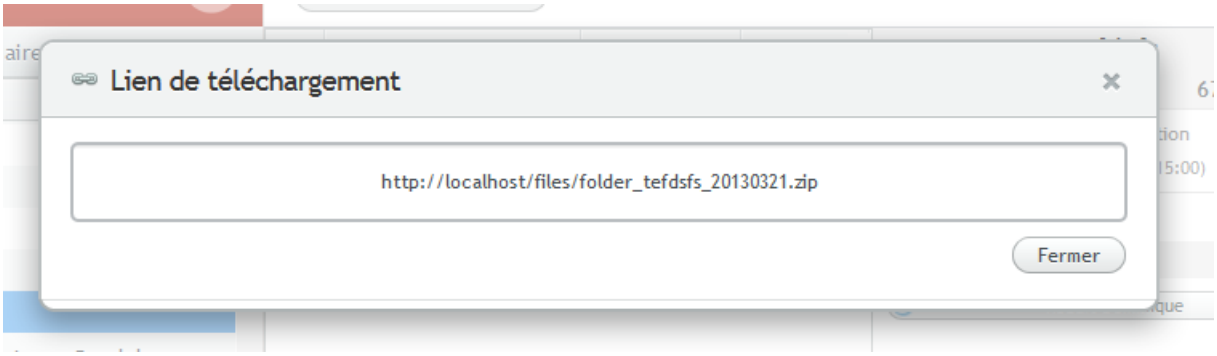


Cette action du menu contextuel n'est disponible que pour les dossiers qui ne sont pas archivés.

L'e-mail reçu est exactement le même qui est envoyé lors de la création du dossier. Par contre, le nombre de jour restant avant l'expiration du dossier est recalculée par rapport à la date d'envoi (et plus par rapport à la date de création du dossier).

11.9. RÉCUPÉRATION DES LIENS

Cette fonction permet de récupérer le lien direct vers un fichier, ou tout simplement vers l'archive ZIP du dossier qui contient la totalité des fichiers.

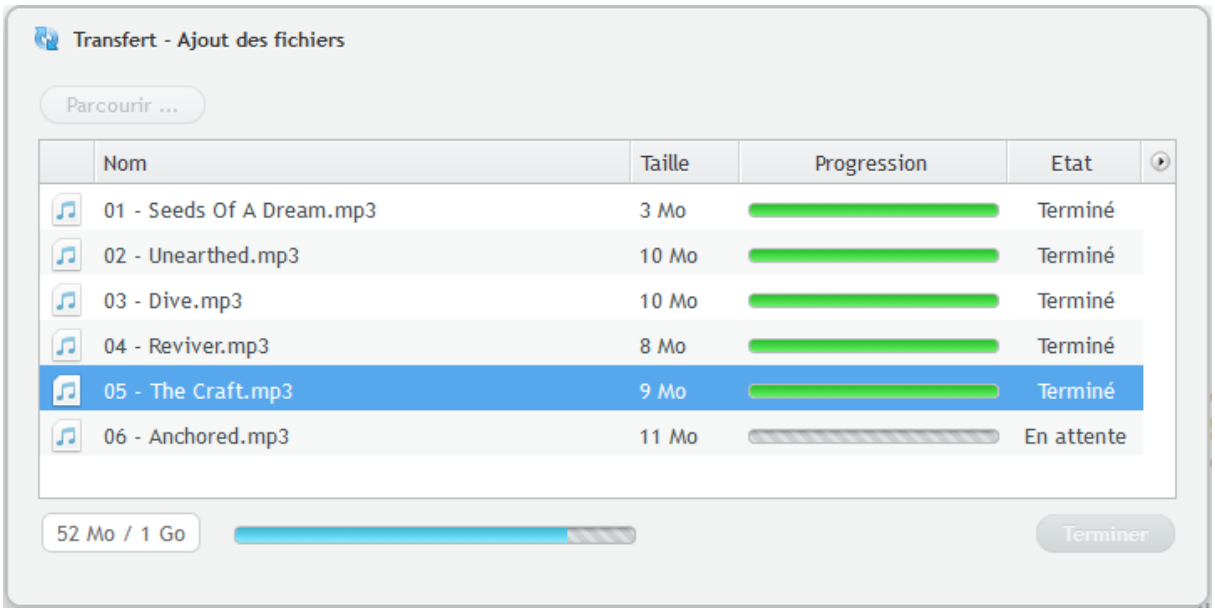


Si un utilisateur souhaite partager un lien pour le téléchargement direct d'un fichier, ou l'archive ZIP du dossier, il lui est très simple de le faire en choisissant l'option « Récupérer le lien » du menu contextuel affiché au clic droit sur ses fichiers et dossiers.

Ce lien permet d'accéder directement au fichier concerné et non à la page de téléchargement des fichiers d'un dossier. L'URL est adaptée en fonction de l'environnement. Si je suis sur le serveur de production, elle changera automatiquement.

12. UPLOAD DES FICHIERS

L'objectif principal de ce projet est, comme son nom l'indique, le transfert de fichier. Il est évident selon moi que cette tâche doit être la plus développée, la plus travaillée, et aussi celle qui devrait apporter le plus de difficultés techniques.



Cette capture représente la fenêtre permettant le transfert des fichiers. Les fichiers sélectionnés par l'utilisateur sont placés dans le tableau et peuvent être supprimés ou renommés à la fin de l'envoi. La progression de l'envoi de chaque fichier est représentée par une barre (verte), elle est également affichée en pourcentage dans la colonne « Etat ». Dès que les fichiers sont sélectionnés, la taille totale de tous ceux qui ont étés ajoutés au tableau est calculée et affichée dans la zone blanche au fond à gauche. Cette zone fait office d'indicateur pour la limite de la taille d'envoi, ainsi, le contact est prévenu avant de sélectionner les fichiers qu'il ne peut pas sélectionner plus de 1 Go.

La barre de progression bleue affiche la progression totale de l'envoi.

12.1. CHOIX DES OUTILS UTILISÉS

EasyUploads 5.1

Que ce soit pour ce projet, mais déjà bien avant, je me suis longtemps documenté sur les manières d'envoyer des fichiers en Java. Je savais très bien le faire en PHP, mais je n'ai jamais trouvé de script java tout prêt qui permettait le transfert de fichiers sans avoir à effectuer plusieurs configurations compliquées. De plus, je n'avais jamais trouvé de solution pour l'envoi de fichiers multiples.

Grâce à Vaadin, qui utilise le Google Web Toolkit pour pouvoir compiler en Java des instructions JavaScript, j'ai enfin pu accéder à cette possibilité. Vaadin propose un composant simple pour l'envoi de fichier, mais je ne le trouvais pas adapté à mes besoins.

La communauté Vaadin propose une multitude d'« add-ons ». Les add-ons sont des composants nouveaux ou basés sur des composants natifs de Vaadin. Ils sont développés et mis à disposition soit par l'entreprise Vaadin elle-même, soit par des personnes ayant une connaissance très poussée dans ce domaine. Les add-ons sont disponibles en téléchargement sur le site officiel de Vaadin et peuvent être modifiés pour personnaliser leur utilisation, mais il est très délicat de le faire car nous n'avons plus à faire à du Vaadin, mais au compilateur GWT, qui est presque un « langage » à part, que je ne maitrise pas.

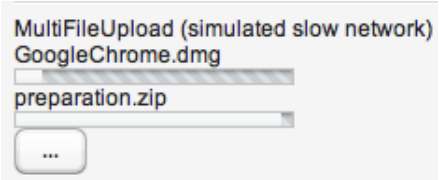
Un add-on nommé « EasyUploads » répondais presque parfaitement à mes besoins. Il permet de l'envoi multiple, il est très simple à utiliser, la progression peut être affichée et gérée très simplement… Je dis « presque », car beaucoup de fonctionnalités n'étaient pas implémentées. Par exemple, je ne pouvais pas détecter lorsqu'un fichier était sélectionné, où quand un envoi était terminé. Tous ces évènement primordiaux pour l'utilisation que j'allais en faire n'étaient pas implémentés.

Heureusement, Vaadin dispose d'une large communauté et le support est très efficace. J'ai décidé d'écrire sur le forum dédié à l'add-on pour exposer mon problème et mes besoins. Deux jours plus tard une réponse m'est parvenue d'un dénommé ***Steffen Ewert***. Il avait ajouté les fonctionnalités dont j'avais besoin et mis à disposition la nouvelle version de l'add-on sur un serveur SVN (Subversion) de Google.

Dans cette nouvelle version, il a implémenté trois nouveaux évènements :

- fileUploadStarted — Déclenché au démarrage du transfert d'un fichier
- fileUploadFinished — Déclenché lorsque l'envoi d'un fichier est terminé
- fileUploadError — Déclenché lorsqu'une erreur de transfert survient

Voici un aperçu de l'add-on de base *(capture tirée de la démo de l'add-on)*



12.2. CONFIGURATION TOMCAT

Pour s'assurer que le serveur soit configuré de manière à accepter l'envoi de gros fichier, notamment que la session n'expire pas si le temps de l'envoi est très long et que la taille d'un gros fichier ne déclenche pas d'exception à la fin d'un téléchargement, des modifications doivent être apportées au niveau de la configuration du serveur Tomcat.

Voici un descriptif des modifications de configurations apportées au connecteur spécifié dans le fichier **server.xml** du répertoire **/opt/tomcat7/conf/.**

Configuration

Paramètre	Valeur	Description
maxPostSize	0	Supprime la limite de taille des requêtes POST
disableUploadTimeout	true	Autorise le serveur à allonger le `connectionTimeout` pendant l'exécution d'une servlet et donc de laisser plus de temps à la servlet pour être exécutée lors de l'envoi d'un gros fichier. Dans le cas d'un fichier de taille standard, le timeout est porté à la valeur spécifie par le paramètre `connectionTimeout` (par défaut : 20 sec.)

Résultat

```
<Connector … maxPostSize="0" disableUploadTimeout="true" />
```

12.3. PERSONNALISATION DE L'ADD-ON

J'ai eu besoin de personnaliser, d'améliorer et d'adapter l'add-on à mes besoins. Comme l'add-on est un fichier JAR inclus dans le projet, je ne peux pas modifier le code source. Lorsque je le fais, je dois recompiler le projet avec le compilateur de Google Web Toolkit. Pour pouvoir développer beaucoup plus simplement mon application sans recompilation, j'ai simplement créé un package ou j'ai copié les classes principales de l'add-on. Tout en conservant les dépendances. Ainsi, je peux modifier à ma convenance le composant principal. C'est la classe `MultipleFileUpload` qui s'occupe de créer ce composant. La classe `CustomUpload` hérite de ce composant et étends ces propriétés. C'est elle qui gère les processus déclenchées par les évènements.

12.3.1. MISE À JOUR VIA LE SERVEUR SVN

L'add-on est souvent mise à jour par différents membres de la communauté Vaadin. Cependant, ce n'est que lors de modifications majeures qu'une nouvelle version est publiée en téléchargement sur le site officiel.

Pour obtenir toutes les dernières mises à jour de l'add-on, il est possible de récupérer directement le code source depuis un serveur SVN de Google.

Pour télécharger la dernière version, il faut disposer d'un client SVN qui se chargera de télécharger les sources sur le serveur. J'ai suivi les recommandations de Monsieur Steffen Ewert en téléchargeant le plugin « Subclipse » pour éclipse.

Pour télécharger ce plugin : Help > Eclipse Marketplace > Rechercer « Subclipse » > Install

12.3.2. INTERFACE D'ÉVÈNEMENTS PERSONNALISÉS

Dans le package `common.component.upload`, se trouve une interface `UploadActionListener`. En Java, une interface donne des « devoirs » aux classes qui l'implémentent. Cela signifie que la classe qui implémente l'interface `UploadActionListener` doit obligatoirement utiliser les méthodes de cette dernière. Une interface est comparable à une classe n'ayant que des méthodes abstraites.

L'interface UploadActionListener définit les actions (ou évènements) qui seront déclenchés lors de l'upload des fichiers.

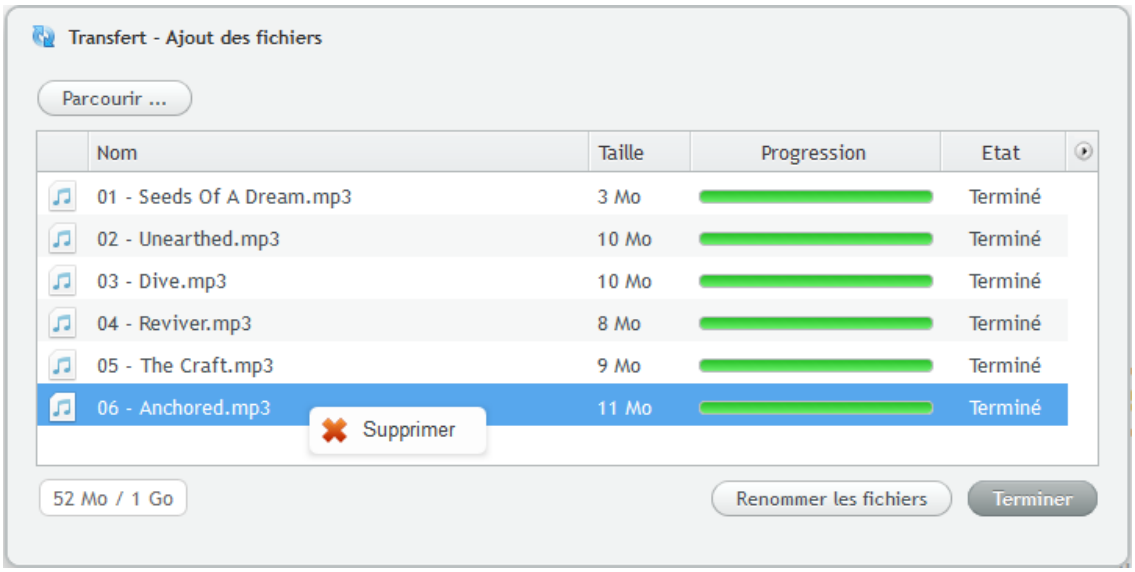
J'ai mentionné au début de ce chapitre les modifications apportées par Monsieur ***Steffen Ewert*** sur l'add-on EasyUploads. C'est dans cette interface qu'il a défini les différents évènements, c'est pourquoi vous y trouverez son nom ainsi que des commentaires en anglais.

Les améliorations apportées par Mr. Ewert étaient très utiles. Toutefois il me manquait quelques évènements pour que l'add-on devienne beaucoup plus flexible d'utilisation.

J'ai analysé le code ajouté par Mr. Ewert et me suis basé sur le même principe pour développer mes propres évènements. J'ai ajouté :

- fileUploadProgress — Déclenché à chaque fois que la progression augmente
- onSelectedFiles — Déclenché juste après la sélection des fichiers
- fileUploadComplete — Déclenché lorsque tous les fichiers sont envoyés

Cela me permet de pouvoir déterminer depuis la classe `CustomUpload` quels fichiers ont été sélectionnés, de pouvoir récupérer la progression de l'envoi, etc… et d'afficher tout cela dans le tableau :



12.3.2.1. CODES D'ERREURS REÇUS À LA SÉLECTION DES FICHIERS

La méthode exécutée lors de la sélection des fichiers (onSelectedFiles) reçoit en paramètre un code d'erreur. Si ce code d'erreur équivaut à 0, il n'y en a pas, sinon voici à quoi correspondent les différents codes d'erreurs :

1 = (Drag&Drop) Certaines données déposées ne sont pas des fichiers.
2 = Il y a des fichiers dont l'extension n'est pas autorisée
3 = La taille des fichiers dépasse la taille maximale autorisée.

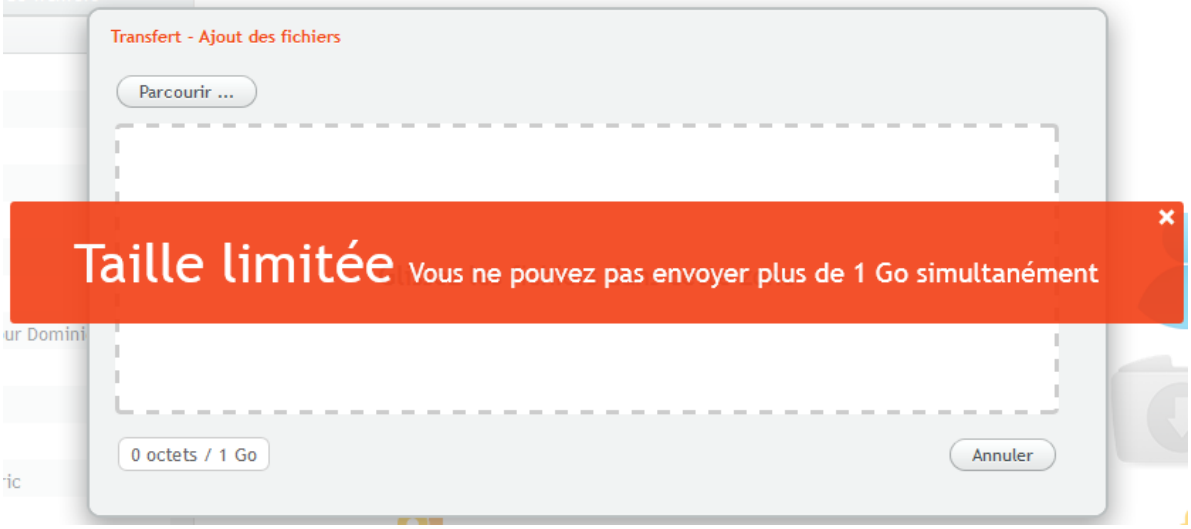
12.3.3. LIMITATION DE LA TAILLE DE LA QUEUE D'ENVOI

L'utilisateur ne doit pas avoir le droit de transférer des fichiers plus volumineux que 1 Go. La taille totale de tous les fichiers qu'il envoie ne doit pas non plus dépasser cette capacité. Pour ce faire, j'ai une fois encore eu recours à la modification de la classe `MultipleUpload`. Chaque fois qu'un fichier est envoyé, sa taille est stockée dans une variable. Chaque taille est additionnée et si la taille totale dépasse 1'073'741'824 octets, les fichiers ne sont pas envoyés. Un message d'avertissement est affiché.

La taille limite autorisée est également un paramètre que l'administrateur peut modifier dans la table `trans_app_param` de la base de données.

Un problème se pose si l'utilisateur choisi un fichier trop volumineux (environ > 2 Go). Ce problème vient du fait que l'add-on stocke la taille dans une variable de type `int` qui ne peut contenir que des valeurs dans un intervalle de -2'147'483'648 à 2'147'483'647. Or, 2 Go = 2'147'483'648 octets. Dans ce cas, on se retrouve donc en dépassement de valeur, ce qui déclenche une erreur.

Pour pallier à ce problème, il faut modifier tous les types de variables qui stockent la taille des fichiers reçus (de type `int` vers `long`). La modification du type de la variable impliquera un temps d'envoi plus long, car chaque fois que des calculs seront faits, ils seront stockés sur 64 bits au lieu de 32 (notamment le calcul de la progression qui se base sur la taille du fichier).



12.3.4. EXTENSIONS AUTORISÉES

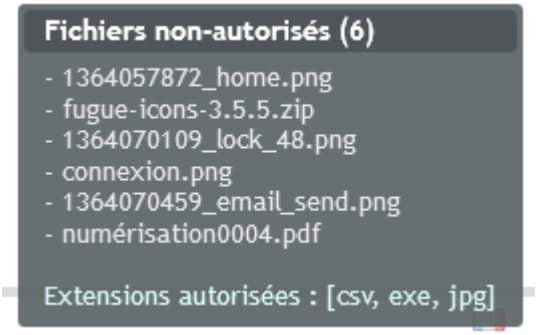
La limitation des extensions autorisées n'était pas un point stipulé dans le cahier des charges. Cependant, Monsieur Lorétan m'a demandé en regardant mon application si j'avais ajouté cette possibilité dans les paramètres. Je l'ai donc implémentée.

Il suffit de modifier le paramètre « `UPLOAD_ALLOWED_EXTENSIONS` » de la table `trans_app_param` dans la base de données, pour déterminer quels types de fichiers peuvent être envoyés.

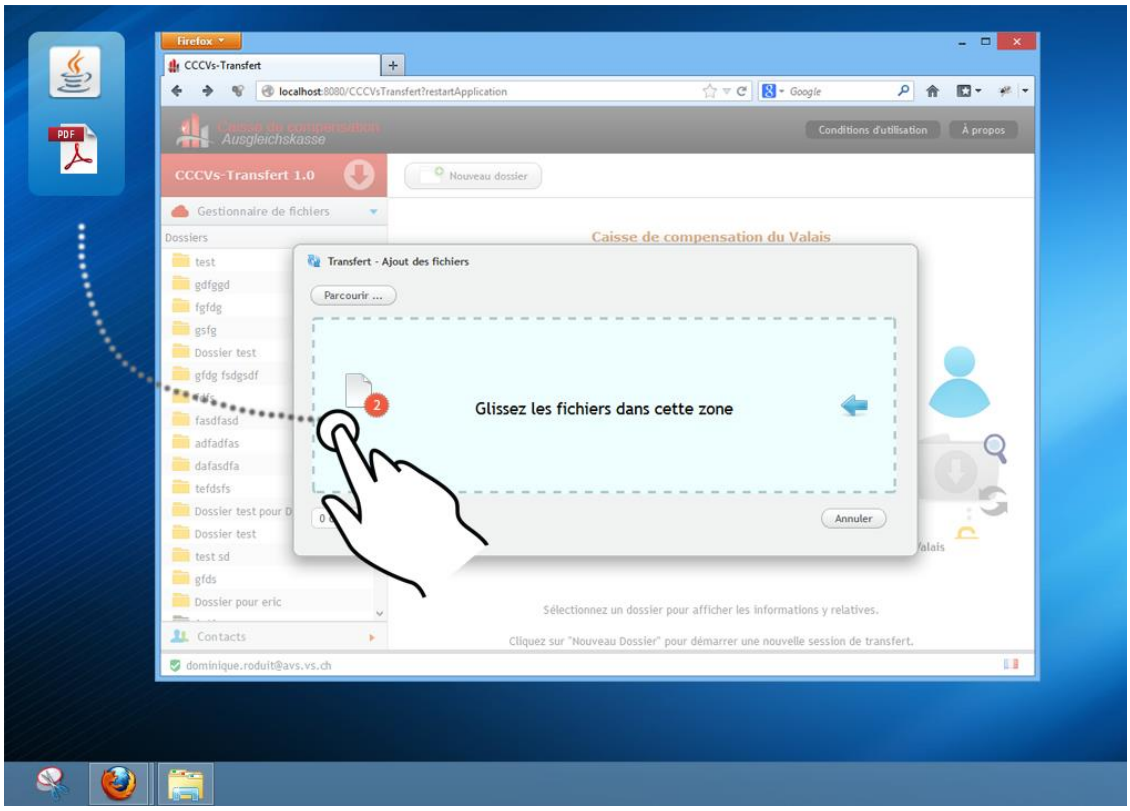
Voici deux exemples de configuration du paramètre :

- exe, csv, xls, txt — Seul les fichiers de type exe, csv, xls et txt sont autorisés
- * — Tous les types de fichiers sont autorisés (par défaut)

Voici un exemple de notification affichée si le paramètre est définit à « csv, exe, jpg » :



12.3.5. DRAG&DROP



Cette zone permet de glisser-déposer les fichiers du système d'exploitation directement dans la fenêtre du navigateur. Celle-ci est implémentée en JavaScript et n'est affichée que sur les navigateurs récents qui intègrent cette technologie. Voici la liste des navigateurs concernés :

- Firefox
- Opéra
- Google Chrome

Si l'utilisateur utilise l'application avec un autre navigateur, la zone sera remplacée par un bloc demandant simplement de sélectionner les fichiers.

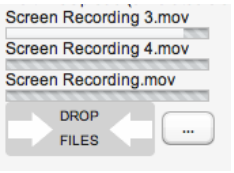
Pour modifier l'aspect de la zone de drag&drop, il faut modifier la feuille de style CSS « easyuploads.css » localisée dans le répertoire « `WebContent/VAADIN/widgetsets/main.widgetset.CccvstransfertWidgetset` ».

Voici l'aspect que j'ai donné à la zone de drag&drop :

Une fois les fichiers déposés, la zone est remplacée par le tableau d'envoi des fichiers.

Attention ! Chaque fois que l'add-on est recompilé, les fichiers du répertoire `WebContent/VAADIN/widgetsets/main.widgetset.CccvstransfertWidgetset` sont réinitialisés. Par conséquent il convient de sauvegarder la feuille de style CSS dans un dossier à part pour la remplacer chaque fois qu'une compilation réinitialise les fichiers de ce dossier.

Si la feuille de style personnalisée n'est pas remplacée par celle d'origine, la zone de drag&drop sera affichée comme au départ :



La class MultipleFileUpload gère séparément les fichiers ajoutés par drag&drop ou ceux sélectionnés via le bouton « Parcourir… ». Cela implique que le code soit quelque fois dédoublé. Il ne peut pas être séparé dans des méthodes puisque les variables ne sont pas incrémentées et manipulées aux mêmes moments dans les deux cas.

Les fichiers ajoutés dans la zone de drag&drop sont beaucoup plus facilement gérables car je peux effectuer les traitements avant que les fichiers soient envoyés et donc bloquer instantanément les fichiers non-autorisés. Par contre, dans le cas du bouton « Parcourir… », dès que les fichiers sont sélectionnés, ils sont immédiatement envoyés et ce, même si les composants graphiques ne sont pas affichés. Les traitements pour l'analyse des informations des fichiers sont donc effectués tout au long du processus d'envoi. Il est beaucoup plus difficile de gérer la limitation des extensions et tailles des fichiers dans ce cas.

12.4. LOCALISATION DES FICHIERS

Par défaut, en spécifiant dans le code Java que les fichiers doivent s'enregistrer dans le répertoire racine, donc, sur ./ , en local, ceux-ci seront sauvegardés à la racine du dossier de notre environnement de développement. Sur le serveur, ils seront localisés dans le dossier de la jre/bin, soit, sur :

```
/usr/lib64/jvm/java-1_6_0-ibm-1.6.0/jre/bin
```

Je ne souhaite pas sauvegarder mes fichiers dans ce répertoire ! C'est pourquoi j'ai simplement indiqué au programme d'enregistrer les fichiers dans le répertoire temporaire de Tomcat.

```
ROOT_DIRECTORY = new File(System.getProperty("java.io.tmpdir")).getPath();
```

Par défaut, ce paramètre pointe vers la racine du répertoire d'installation de Tomcat :

```
/opt/tomcat7/temp
```

Mais une fois encore, je ne veux pas enregistrer les fichiers dans ce répertoire puisqu'il ne me permet pas de pouvoir les télécharger depuis une URL.

Pour pouvoir les atteindre depuis internet, il faut donc placer les fichiers dans un répertoire du dossier "webapps", qui se trouve lui aussi à la racine du répertoire d'installation de Tomcat. J'ai créé un dossier « files » qui contiendra les fichiers et lui ai donné tous les droits pour voir si je pouvais y télécharger un fichier, puis y accéder.

```
opt/tomcat7/webapps/files/
```

Là, tout va bien. Mais il reste encore à configurer le fichier :

```
catalina.sh         opt/tomcat7/bin/catalina.sh
```

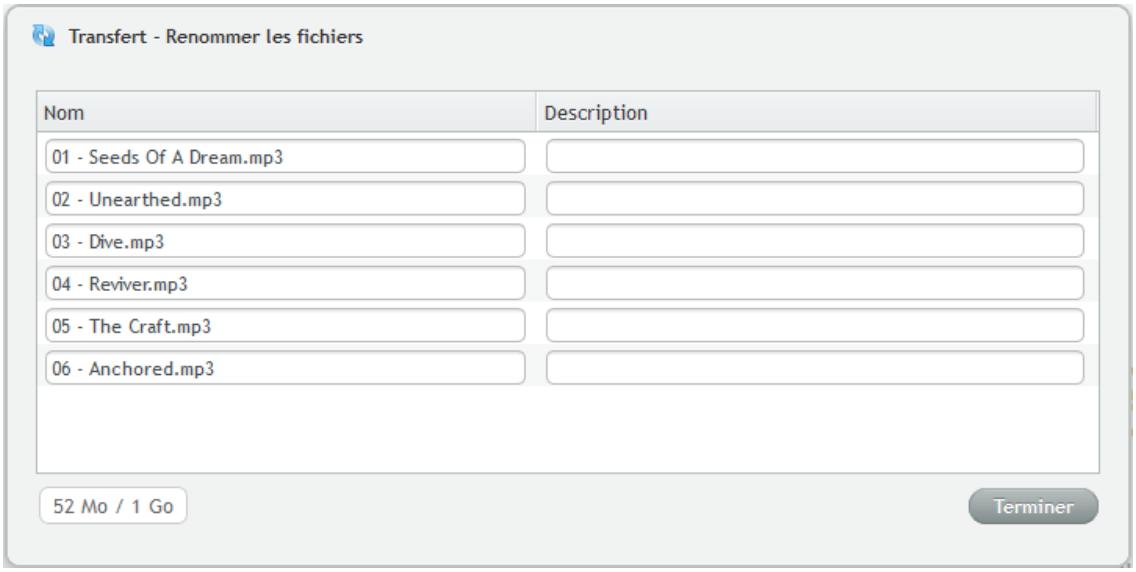
Dans ce fichier, la variable à modifier est :

```
CATALINA_TMPDIR="$CATALINA_BASE"/webapps/files
```

Les fichiers sont maintenant accessibles depuis l'adresse http://10.76.210.172:8080/files/file.x

12.5. RENOMMAGE DES FICHIERS

Lorsque les fichiers ont été envoyés, l'utilisateur peut alors choisir de les renommer et de leur donner une description. Si le fichier est renommé, c'est le nouveau nom spécifié qui sera affiché aux destinataires lorsqu'ils accèderont au dossier. Si les noms ne sont pas modifiés, le nom d'origine sera conservé. La description est facultative. D'ailleurs, l'étape de « renommage » est complètement facultative elle aussi. L'utilisateur peut choisir entre le bouton « Renommer les fichiers » ou « Terminer ». S'il choisit « Terminer », le dossier est envoyé tel quel, avec les noms d'origine.



13. CRYPTAGE DES PARAMÈTRES DE L'URL

Lorsqu'un utilisateur de l'interface achève la création d'un dossier avec succès, un message électronique est envoyé à chacun des destinataires inclus. Cet e-mail contient un lien pour accéder au dossier dans le but de pouvoir télécharger les fichiers contenus. Ce lien comprend des paramètres cryptés pour que la personne ne puisse pas accéder à un dossier dont elle n'a pas accès.

Il en est de même lorsqu'un utilisateur s'inscrit sur l'application. Un mail contenant un lien sécurisé lui permet de valider son compte.

Voici comment se présente un « paramètre » d'URL :

```
https://cccvstransfert.vs.ch/?valid=9dfs6e12c654da9e84fds61c3...&id=d84fs6qs5...
```

Dans cette URL, `valid` et `id` sont les noms des paramètres (les clés) et ce qui vient après le signe '=', la valeur du paramètre en question.

Si l'utilisateur change un seul caractère dans la valeur du paramètre, celui-ci ne sera plus valide et l'application émettra un message d'erreur ou ignorera simplement le paramètre selon les cas.

13.1. ALGORITHME UTILISÉ

J'ai choisi d'utiliser l'algorithme de cryptage SHA-256.

Pourquoi ?

Il est avant tout extrêmement sécurisé puisque théoriquement un résultat serait approximativement trouvé après 2^{128} = 340'000'000'000'000'000'000'000'000'000'000'000 opérations, ce qui est (*pour l'instant*) irréalisable.

Il permet, comme la fonction de hachage SHA-1 de condenser des messages de 2^{64} bits au maximum. Ce qui est bien au-delà de ce que j'utiliserai.

En outre, cette fonction est disponible nativement dans MySQL. C'est d'ailleurs une des raisons qui m'ont poussé à utiliser ce SGBD plutôt que DB2. Il était important que je puisse disposer du même algorithme sur le SGBD que dans mon code Java.

13.2. PROCÉDÉ

Cette section vise à expliquer la méthode que j'ai utilisé pour former les valeurs à crypter et comment j'ai opéré pour ressortir les informations correspondantes aux valeurs des paramètres.

13.2.1. FORMATION DES PARAMÈTRES

Pour construire l'URL des liens cryptés envoyés par mail, j'ai besoin :

- De l'URL de base, sans paramètres (ex. https://cccvstransfert.vs.ch/app/)
- Des noms des paramètres
- Des valeurs des paramètres

Le code pour obtenir l'URL de base est le suivant : `Global.URL`. Il s'agit d'un attribut `static` de la méthode `global.Global` qui est initialisé dans la classe `main.CccvsTransfert`.

Les noms des paramètres sont enregistrés comme paramètres dans la base de données, ils sont donc récupérables de la sorte : `Global.getParm("URL_PARM_VALIDATION")`.

Voici maintenant comment je forme la valeur du paramètre. Je prendrai pour exemple le cryptage du lien envoyé pour la validation des comptes :

Lorsque l'utilisateur clic sur le lien qu'il a reçu, j'ai besoin de pouvoir détecter son adresse e-mail pour l'identifier et effectuer les modifications nécessaires dans la base de données.

Je vais donc crypter son adresse e-mail. Mais je ne souhaite pas passer la valeur cryptée de son adresse e-mail dans l'URL car ce serait beaucoup trop facile pour un informaticien qui connaît le cryptage de pouvoir valider manuellement n'importe quelle adresse. Je vais donc utiliser une sorte de « clé » de cryptage que je vais placer juste devant l'adresse e-mail. Cette clé de cryptage est enregistrée comme paramètre dans la base de données. Elle est récupérable comme cela : `Global.getParm("SECRET_KEY")`.

Admettons que la valeur de la clé de cryptage soie « `aimeJava` ». L'utilisateur « `dominique.roduit@avs.vs.ch` » s'inscrit sur l'application. La valeur du paramètre sera donc le hachage de la chaine « `aimeJavadominique.roduit@avs.vs.ch` ».

Voici ce que représente cette chaîne, cryptée en SHA-256 :

```
ba55dcc25b2f3c3e172d97003aed2bb1992089c6b94394b103c4ebbee4cd427e
```

En associant les parties cités ci-dessus, voici comment le paramètre est formé :

```
Global.getParm("URL_PARM_VALIDATION")+"="+SHA256.getHashValue(Global.getParm("SECRET_KEY")+to)
```

13.2.2. RÉCUPÉRATION DES VALEURS

Je reçois dans l'URL le paramètre suivant :

```
?valid=ba55dcc25b2f3c3e172d97003aed2bb1992089c6b94394b103c4ebbee4cd427e
```

Je souhaite connaitre quel est l'utilisateur qui possède l'adresse e-mail contenue dans ce paramètre crypté. Voici comment je m'y prends :

Cette fois-ci, c'est une requête SQL qui intervient :

```
SELECT * FROM tbl_users
WHERE
SHA2(CONCAT('"+Global.getParm("SECRET_KEY")+"', user_mail), 256)="+
Utilities.formatSQL(URLParameter)
```

`URLParameter` correspond à la valeur du paramètre reçu.

Explication de la requête :

Sélectionne tous les champs de la table des utilisateurs lorsque la valeur du paramètre reçu correspond à la valeur hachée de la concaténation de la clé suivie de l'adresse e-mail.

Cette requête me retourne bien l'entrée correspondante à l'utilisateur. Je peux stocker les informations sur ce dernier et mettre à jour la base de données, l'adresse est validée.

14. TÉLÉCHARGEMENT DES FICHIERS

14.1. CRYPTAGE DES PARAMÈTRES URL

Dans le l'URL du lien reçu par mail, 3 paramètres sont cryptés :

- download Identifiant du dossier auquel on accède
- id Identifiant du contact qui accède au dossier
- idm Adresse e-mail du contact qui accède

Si un des 3 paramètres est modifié d'un seul caractère, le lien ne sera plus valide. Un message informant de l'invalidité de l'URL est affiché.

Pour se rendre compte de la solidité de l'algorithme de cryptage, voici un aperçu concret d'une URL reçue par mail pour accéder à un dossier :

http://localhost:8080/CCCVsTransfert/?restartApplication&download=de944d212f396b04f7b0351ccfa420ead35706863e1f77fba6d28716c04429e3&idm=3930cae83d05034151b729e3c7be2e0539d28d7d4fb469ed39c10c8f2118317d&id=e1bb9d9ee8aaa5dfcf9c54e4f16ff7d52fef57c09e822ebfe260cf48e7c58659

14.2. FICHIERS AFFICHÉS DIRECTEMENT DANS LE NAVIGATEUR

Lorsqu'une image ou un fichier PDF est téléchargé, ceux-ci s'affichent directement dans le navigateur du client sans qu'une fenêtre demande où le fichier doit-être enregistré.

Pourquoi ?

Car ceci ne dépend pas de l'application mais de la configuration du client.

Par exemple, pour choisir de télécharger les fichiers PDF au lieu de les afficher dans le navigateur, il faut modifier les préférences du lecteur de PDF. Avec Adobe Reader, la configuration se fait dans « Edition > Préférences > Internet ».

Cependant, j'ai la possibilité de forcer le navigateur à afficher la fenêtre de téléchargement mais je n'ai pas souhaité le faire car le fait que le fichier PDF soit affiché directement dans le navigateur n'est pas gênant et l'utilisateur peut toujours choisir de télécharger ce fichier sur son bureau.

15. SUPPRESSION AUTOMATIQUE DES FICHIERS

La suppression automatique des fichiers d'un dossier implique que celui-ci ne soit plus accessible. Un tel évènement survient lorsque la durée de vie du dossier (de 1 à 10 jours) expire.

Ce n'est pas l'interface utilisateur qui doit vérifier lesquels dossiers seront bientôt archivés, ni envoyer les mails pour alerter les utilisateurs de cette action. Cette tâche sera confiée à la crontab du serveur Linux. Ou plus exactement, la crontab se chargera d'exécuter un deuxième projet, qui lui-même se chargera d'exécuter les requêtes nécessaires. Ce projet doit posséder ses propres classes, ses propres librairies, ses propres configurations,… Il doit être complètement indépendant de l'application utilisateur car il sera exécuté par le système et n'a aucun intérêt à être pourvu d'une interface graphique.

Un nouveau projet Java a été créé pour effectuer toutes les tâches automatiques. J'ai nommé ce projet « `RemoveArchives` » (puisque son rôle sera principalement de supprimer les fichiers et d'archiver les dossiers). Il inclut les librairies `JavaMail` et `Connector/J`.

Ce projet utilise plusieurs copies des classes que j'ai créées pour l'interface utilisateur. Entre autre, la classe de connexion à la base de données et celle pour l'envoi des mails.

15.1. EXPORTATION DU PROJET

L'exportation du projet (.jar) pour la suppression automatique des fichiers a été un problème relativement compliqué qui a demandé beaucoup de temps. Au départ, je ne connaissais rien de l'exportation d'un fichier Jar. A vrai dire, je ne savais même pas si c'était le format qu'il me fallait pour être exécuté automatiquement par la crontab, mais j'avais l'intuition que je devais utiliser ce format. J'ai donc fais des recherches sur internet et j'ai suivi à la lettre les instructions pour l'exportation d'un projet JAR. Mon fichier MANIFEST.MF était à première vue correct, tout était exporté correctement et aux bons endroits dans l'archive JAR générée, mais rien n'y faisait. Une fois exécuté, je réceptionnais toujours le même message à l'écran :

```
java.lang.ClassNotFoundException com.mysql.jdb.Driver
```

Ce message est clair, il m'indique que j'utilise une class qui n'est pas trouvée. Pourtant la librairie du Connector/J se trouvait dans mon répertoire et dans le classpath du fichier Manifest.

J'ai tenté de :

- Définir les ressources dans l'option -classpath de la commande d'exécution JAR
- Intégrer une copie de mes librairies à la racine du répertoire d'installation des librairies natives de Java.
- Dépaqueter les archives des archives dont j'avais besoin pour les importer directement dans les sources de mon projet.

Aucune des solutions expérimentées ci-dessus n'a fonctionné.

Après de longues recherches sans résultat, j'ai demandé de l'aide à Mathieu Bérard, très expérimenté dans le domaine de la programmation Java. Il est venu m'épauler et nous avons essayé de favoriser une option différente pour l'exportation du projet JAR depuis Eclipse. Nous avons choisi "JAR Exécutable" au lieu de "JAR".

Là, le problème était résolu. Je ne saurais jamais pourquoi en choisissant le format "JAR" simple il ne trouvait pas les librairies incluses car elles étaient pourtant clairement spécifiées dans le classpath de mon fichier MANIFEST.

Voici la marche à suivre ainsi que les configurations pour l'exportation du projet au format Jar exécutable.

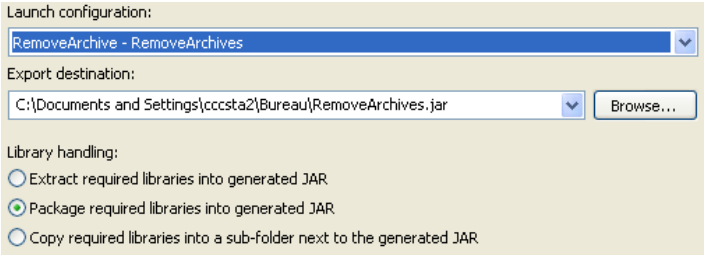
Sur Eclipse

Class main exécutée au lancement du JAR

- Clic droit sur le projet > Run as > Run configurations …
- Clic droit sur « Java Application » dans le menu de gauche > New
- Dans « Main class », renseigner la class contenant la méthode main à exécuter

Compiler le projet en archive JAR

- Effectuez un clic droit sur votre Projet > Export …
- Sélectionnez "Runnable JAR file" (Fichier JAR exécutable)
- Sélectionnez la configuration créée ci-dessus



Il est important bien entendu de renseigner comme dans tout projet Java le chemin vers les librairies dans le « Java build path » pour que celles-ci puissent être empaquetées correctement dans le fichier JAR compilé.

15.2. EXÉCUTION D'UN JAR COMPILÉ

Un fichier JAR est une archive compressée contenant des ressources Java. Par conséquent, il est très simple d'ouvrir ce type de document à l'aide les logiciels standard de décompression. Les fichiers JAR peuvent également être représentés comme des exécutables (.exe) qui seraient créés en C++, à l'exception qu'ils ont l'avantage d'être multiplateforme, c'est-à-dire qu'ils peuvent être exécuté sur tous les systèmes d'exploitation sur lesquels une machine virtuelle Java (JVM) est installé. Ce format à cependant le désavantage d'être plutôt méconnu du grand public, ce qui rend les utilisateurs plus méfiants au téléchargement et à l'exécution. Heureusement, des logiciels existent pour créer un exécutable qui se chargera uniquement d'exécuter lui-même le document JAR.

Dans notre cas, nous n'utiliserons pas une archive JAR à des fins d'exécution manuelle avec une interface graphique, nous n'aurons donc pas à nous soucier de l'exécutable. Nous souhaitons un fichier qui puisse être lancé automatiquement depuis un serveur Linux.

Voici la commande pour l'exécution d'un JAR en mode console dans un environnement Linux :

1 – Se déplacer dans le dossier où se trouve le fichier JAR (cd …)
2 – Exécuter la commande `java -jar NomDuFichier.jar`

15.3. AUTOMATISATION DE L'EXÉCUTION

Pour automatiser l'exécution du projet JAR, il a fallu utiliser la `crontab` de Linux.

La `crontab` est un outil qui permet de programmer l'exécution régulière d'un script quelconque. Contrairement à `at` qui n'exécutera le programme qu'une seule fois, `crontab` permet de faire en sorte que l'exécution soit répétée : toutes les heures, toutes les minutes, tous les jours, tous les trois jours, etc. *(Source de la description : http://bit.ly/XTMy0Q)*

Ma commande sera exécutée depuis un fichier shell (.sh). De cette manière, je pourrai le modifier à ma convenance sans jamais toucher à la configuration de la `crontab`.

J'ai décidé de placer mon fichier shell dans le répertoire /opt/tomcat7/work/CCCVsTransfert-cron. Il contient la commande `java -jar /opt/tomcat7/work/CCCVsTransfert-cron/RemoveArchives.jar` expliquée ci-dessus.

Comment utiliser la `crontab` ?

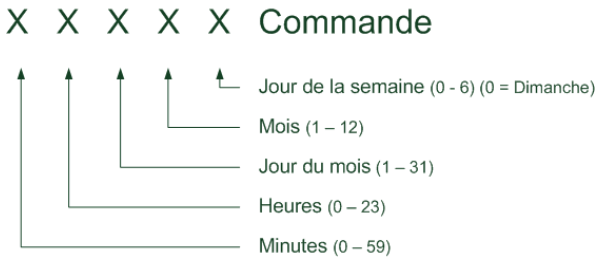
Voici les deux paramètres les plus importants à connaitre :

- `crontab -e` Modifier la crontab
- `crontab -l` Afficher les entrées actuelles de la crontab

Au départ, la crontab n'est pas encore créée. Vous noterez qu'il y a une crontab par utilisateur.

Comment configurer l'exécution régulière d'une commande ?

Chaque ligne du fichier correspond à une commande. Voici un schéma qui résume la syntaxe d'une de ces lignes :



Source : *http://bit.ly/XTMy0Q*

Pour programmer l'exécution du projet JAR toutes les heures de chaque jour de l'année, j'ai donc modifié la crontab à l'aide de la commande `crontab -e`, puis entré la ligne suivante au début du fichier :

```
0 * * * * /opt/tomcat7/work/CCCVsTransfert-cron/jar-launscher.sh
```

Comment savoir que `crontab` a exécuté le script shell et que la commande à fonctionnée ?

Les logs de la crontab sont enregistrés dans /var/mail/root. J'ai donc utilisé la commande Java `System.out.print("texte");` pour que toutes les commandes exécutées dans le JAR (Suppression des fichiers, mises à jour de la base de données, …) soient enregistrées dans ce fichier de log. Pour vérifier le bon fonctionnement du script, j'ai d'abord effectué des tests en programmant l'exécution du script chaque minute dans la `crontab`.

Aperçu du résultat affiché lors de l'exécution du projet JAR

```
prdWTransfert:/opt/tomcat7/work/CCCVsTransfert-cron # ./jar-launscher.sh
-- CCCVs-Transfert -- Lancement du script de suppression des dossiers
2013-03-22 11:21:08.820  -- [Connexion rÃ©ussie] localhost/db_cccvstransfert
2013-03-22 11:21:08.825  -- [OK] > SELECT * FROM trans_app_param
2013-03-22 11:21:08.828  -- [OK] > SELECT * FROM trans_folders LEFT JOIN trans_users ON
PKNoUser=FKNoUser WHERE folder_expiration<ADDDATE(now(),1) AND DATEDIFF(folder_expiratio
n,folder_creation_date)>1 AND DATEDIFF(folder_expiration,now())>0 AND folder_archive=0
2013-03-22 11:21:08.829  -- [OK] > SELECT * FROM trans_folders LEFT JOIN trans_users ON
PKNoUser=FKNoUser WHERE folder_expiration<=now() AND folder_archive=0
Aucun dossier a archiver
```

16. ADMINISTRATION

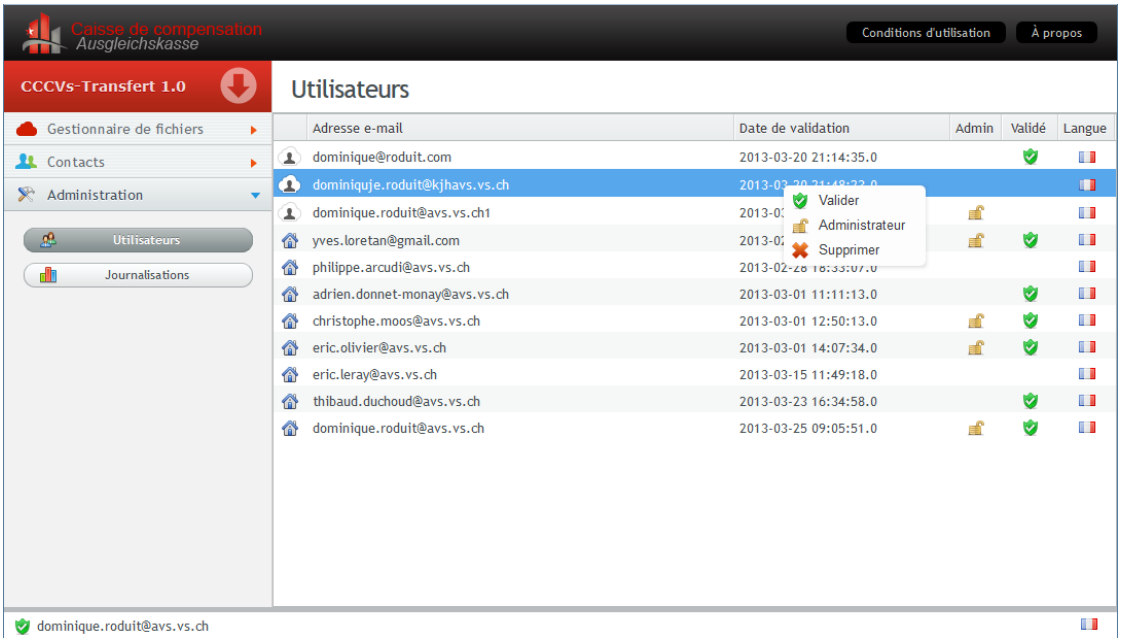
Le cahier des charges expose deux parties distinctes. La première, qui est la partie obligatoire, celle que je dois réaliser, et la deuxième, qui est facultative et que je ne devais faire que lorsque toutes les tâches de la première parties étaient accomplies.

Or, il se trouve que toute la première partie a été effectuée. J'ai longuement cherché les erreurs et amélioré les fonctionnalités de cette partie du mieux que j'ai pu, mais à un moment donné, il se trouve que je ne savais plus quoi ajouter pour améliorer cette partie, j'ai alors décidé de commencer la 2e partie du cahier des charges, 3 jours avant la fin du projet. Je n'imaginais pas du tout pouvoir terminer cette partie en si peu de temps. Cependant, j'ai réussi à quelques détails près à la réaliser complètement.

Ces quelques détails sont l'intégration de graphiques pour la présentation des données journalisées. Au moment d'élaborer le cahier des charges, je ne sais pas à quoi j'ai pensé en faisant allusion à la représentation de données sous forme graphique mais au moment de réfléchir à la conception de cette page, je ne voyais pas quelles données utiles je pouvais représenter, si ce n'était des sommes et moyennes sur les actions effectuées, ce qui n'aurait pas apporté grand-chose. Je n'ai donc pas intégré de graphique mais j'ai simplement décidé de créer une page ou l'administrateur pourrait consulter toutes les actions journalisées.

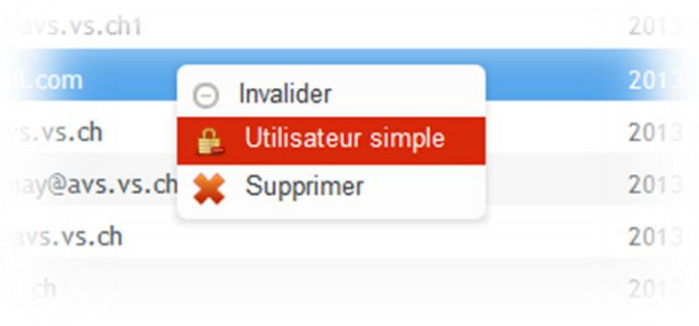
L'administration des données implique des utilisateurs aux pouvoirs d'administrateurs. La partie 2 du cahier des charges spécifiait qu'il serait intéressant de créer une interface à part uniquement accessible aux administrateurs. J'ai nuancé ce propos en intégrant directement la zone d'administration à l'interface utilisateur de l'application. Son menu latéral se prêtait très bien à l'ajout d'un nouvel onglet. Il a également fallu que j'apporte une modification à la base de données en ajoutant un champ « user_admin » dans la table des utilisateurs pour distinguer les administrateurs des simples utilisateurs.

16.1. UTILISATEURS



Cette page est un module de gestion des utilisateurs de l'interface. Toutes les personnes inscrites, qu'elles aient validées leur compte ou non sont affichées dans cette table.

L'administrateur peut décider de propager ses pouvoirs à d'autres utilisateurs via l'option « Administrateur » du menu contextuel. Ce menu est dynamique. Lorsqu'il point sur un administrateur qui a validé son compte, il proposera les actions inverses :



Le défi de cette page, qui lorsqu'on le voit peut paraître simple, était de présenter les données de manières claires et intuitives. Pour cela, j'ai remplacé le texte par des icônes. Celles-ci me semblent beaucoup plus parlantes dès le premier coup d'œil.

Chaque fois qu'une action du menu contextuel est exécutée, la cellule concernée par l'action est instantanément mise à jour. Par exemple, si on choisit de retirer les droits d'un administrateur, la cellule qui contient l'icône du cadenas ouvert disparaitra directement. La base de données est bien entendue mise à jour en parallèle, mais le tableau n'est jamais rechargé, pour éviter de surcharger la mémoire avec des requêtes inutiles. Seul le cache du tableau est rafraichi.

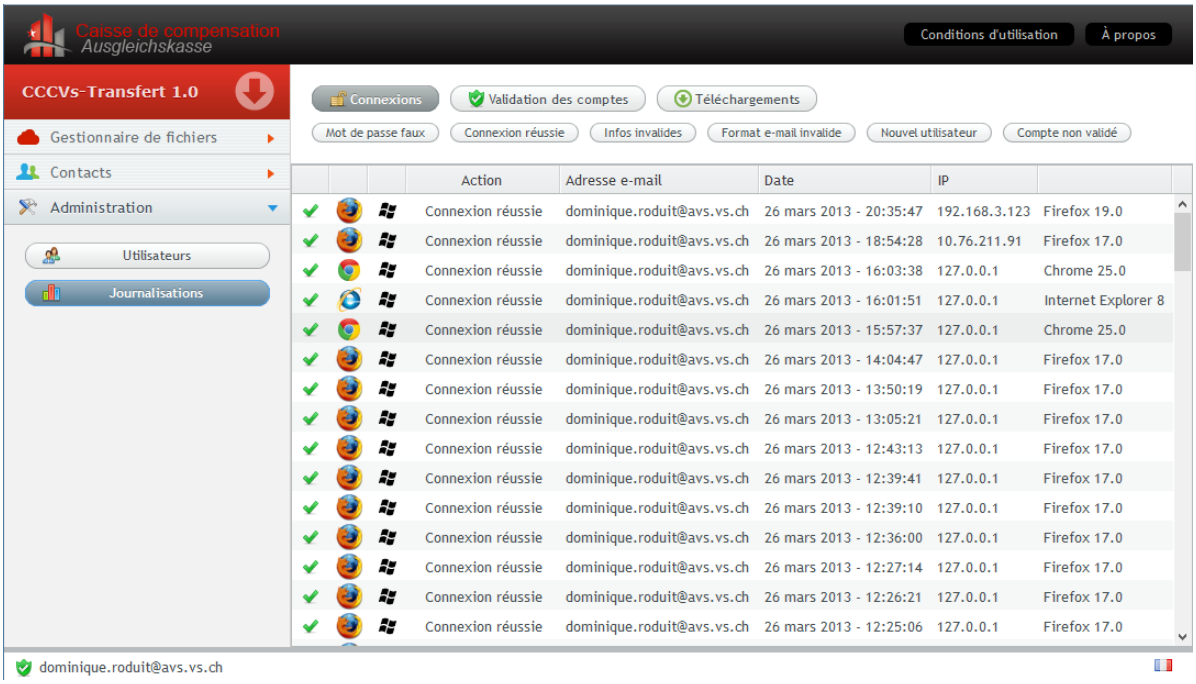
Comme le montre les illustrations ci-dessus, le menu contextuel propose trois actions :

- Valider / Invalider
- Administrateur / Utilisateur simple
- Supprimer

La suppression d'un utilisateur est effectuée en cascade sur tous les enregistrements qui lui sont liés. C'est-à-dire que tous les dossiers, tous les contacts et toutes les données journalisées à son propos seront instantanément supprimées en même temps que son compte. S'il veut pouvoir accéder à nouveau, il devra se réinscrire.

Je n'ai pas implémenté le fait de pouvoir bannir des utilisateurs en fonction de leur adresse IP publique mais il serait intéressant de réaliser cette amélioration qui pourrait nous permettre de bloquer une personne utilisant le service à des fins inadaptées.

16.2. JOURNALISATIONS



Cette page de journalisation n'est qu'une page d'affichage et de filtrage d'informations. Elle ne permet pas l'édition de quelque donnée que ce soit, par conséquent, il n'y a pas de menu contextuel affiché sur le tableau.

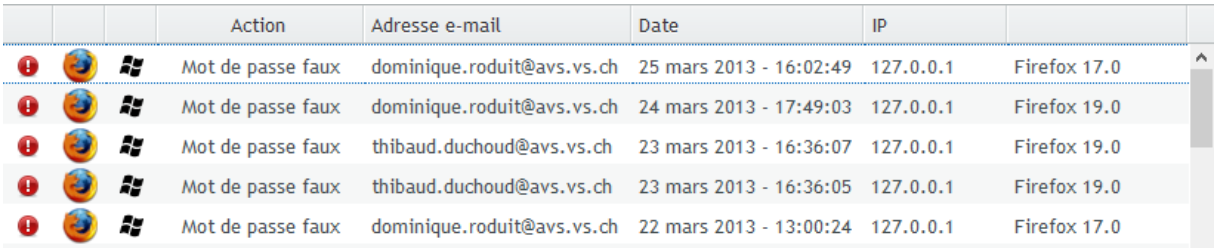
Il regroupe en une seule table les différentes catégories d'actions journalisées à travers l'application.

Dans le ruban, les boutons du haut représentent ces catégories.

Les boutons du dessous sont des « filtres ». Ils permettent de cibler l'information. Par exemple, pour la catégorie « Connexions », voici les différents filtres proposés :

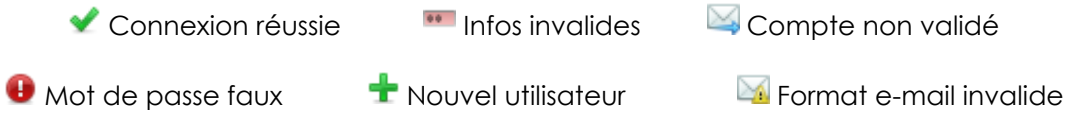
Pour illustrer l'effet de ces filtres, en voici un exemple :

En choisissant le premier filtre, seul les entrées qui ont été journalisées lorsqu'un utilisateur avait saisi un faux mot de passe sont affichées. Voici ce que cela donnerait :



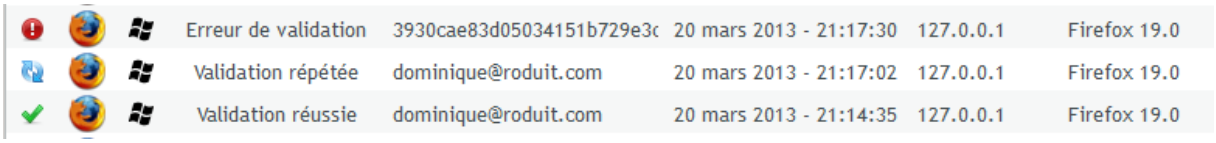
Action	Adresse e-mail	Date	IP	
Mot de passe faux	dominique.roduit@avs.vs.ch	25 mars 2013 - 16:02:49	127.0.0.1	Firefox 17.0
Mot de passe faux	dominique.roduit@avs.vs.ch	24 mars 2013 - 17:49:03	127.0.0.1	Firefox 19.0
Mot de passe faux	thibaud.duchoud@avs.vs.ch	23 mars 2013 - 16:36:07	127.0.0.1	Firefox 19.0
Mot de passe faux	thibaud.duchoud@avs.vs.ch	23 mars 2013 - 16:36:05	127.0.0.1	Firefox 19.0
Mot de passe faux	dominique.roduit@avs.vs.ch	22 mars 2013 - 13:00:24	127.0.0.1	Firefox 17.0

Si aucun filtre n'est sélectionné, toutes les catégories d'actions sont affichées. Chaque type d'action est identifié par sa propre icône. Voici les différentes icônes correspondants aux filtres.



Validation des comptes

Voici les filtres proposés pour la catégorie « Validation des comptes », qui affiche les actions journalisées lorsqu'un utilisateur valide son compte via le lien qu'il a reçu par mail.



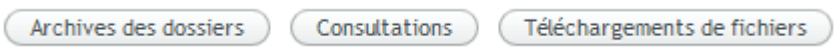
Erreur de validation	3930cae83d05034151b729e3(	20 mars 2013 - 21:17:30	127.0.0.1	Firefox 19.0
Validation répétée	dominique@roduit.com	20 mars 2013 - 21:17:02	127.0.0.1	Firefox 19.0
Validation réussie	dominique@roduit.com	20 mars 2013 - 21:14:35	127.0.0.1	Firefox 19.0

Les erreurs de validation sont déclenchées lorsqu'un utilisateur malintentionné essaie de modifier le paramètre de l'URL. Ce n'est donc plus une adresse e-mail qui est enregistrée mais la valeur du paramètre tel qu'il l'a modifié, nous pouvons ainsi savoir s'il a tenté une injection SQL.

Téléchargements

La catégorie « Téléchargements » propose les actions journalisées lorsqu'un contact télécharge les fichiers d'un dossier, le consulte ou télécharge l'archive de ce dernier.

Les filtres proposés sont les suivants :



Consultation	dominique.roduit@avs.vs.ch	test	-	Dominique Roduit <dominique.r(	26 mars 2013 - 10:53:21
Téléchargement	dominique.roduit@avs.vs.ch	test	Archive du dossier	Dominique Roduit <dominique.r(	26 mars 2013 - 10:53:26
Téléchargement	dominique.roduit@avs.vs.ch	test	putty.e (putty_201303261	Dominique Roduit <dominique.r(	26 mars 2013 - 10:53:30

17. CERTIFICAT SSL

SSL (Secure Sockets Layer)

Tant que l'application est en développement, et durant toute la durée du TPI, il n'est pas demandé d'utiliser un certificat validé par une autorité de certification reconnue, mais simplement d'utiliser déjà une connexion HTTP sur SSL à l'aide d'un certificat, forcément auto-signé et donc valide seulement sur confirmation de l'utilisateur. Ce certificat ne peut pas être reconnu automatiquement pour les navigateurs tant qu'il ne sera pas certifié par une des autorités de certification de la liste blanche.

Sur Mozilla Firefox, nous pouvons trouver cette liste dans le menu

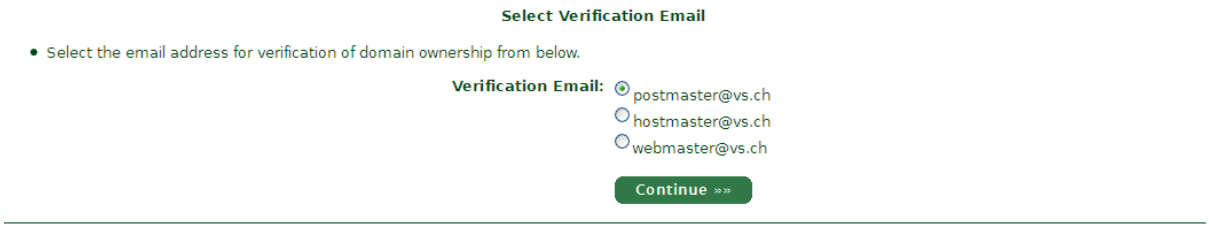
```
Outils > Options > Avancé > Chiffrement > Afficher les certificats > Autorités
```

Sur Internet explorer 8, cette liste se trouve dans

```
Outils > Options internet > Contenu > Certificats > Autorités principales de confiance
```

Peu d'autorités de certification proposent des certificats valides gratuits. Une d'entre elles, nommée "StartSSL" (https://cert.startcom.org/) nous permet de le faire, j'ai donc essayé de profiter d'une autorité de certification valide.

Seulement, plusieurs problèmes se posent dans mon cas puisque pour valider le certificat l'autorité a besoin que je valide mon identité, mon adresse e-mail et que je prouve que je suis bien le propriétaire du nom de domaine sous lequel est hébergée mon application. Cependant, celle-ci est un sous-domaine de l'enregistrement "vs.ch", propriété de l'état du valais, et je ne peux donc pas valider que je suis propriétaire de ce domaine.



Il me faudra donc temporairement utiliser un certificat signé par mes soins.

Pour la génération d'un certificat SSL auto-signé, voici le déroulement des instructions suivies depuis le site officiel de Tomcat, sur la page **http://bit.ly/X4lI1O**.

17.1. WINDOWS

Tout d'abord, un certificat est généré en local. Il est plus simple de le faire d'abord sur Windows (sur le poste de développement), ou le débogage est relativement rapide, puis de répéter les mêmes manipulations sur le serveur par la suite en utilisant les chemins différents.

C'est aussi un moyen de s'assurer qu'il n'y ait pas de différence entre l'environnement de développement et celui du serveur en production.

17.1.1. GÉNÉRATION DU KEYSTORE POUR LE CERTIFICAT

Tomcat ne fonctionne actuellement que sur des fichiers de clés aux formats JKS ou PKCS. L'algorithme de signature que nous allons utiliser pour notre magasin de clé est le PKCS (SHA-1 pour l'algorithme d'emprunte numérique et chiffrement RSA sur 2048 bits pour la clé publique).

Chaque entrée dans un magasin de clé est identifiée par une chaîne d'alias. Alors que de nombreuses implémentations de magasin de clés traitent les alias d'une manière insensible à la casse, ceux du PKCS sont sensibles. Pour éviter des problèmes, il est alors recommandé d'utiliser des alias dont la casse ne diffère pas.

Code console

```
C:\Program Files\Java\jre7\bin>keytool -genkey -alias tomcat -keyalg RSA –
keystore C:\temp\test.keystore

Entrez le mot de passe du fichier de clés : k******* (PAS DE MAJUSCULE !)
Ressaisissez le nouveau mot de passe : k******* (PAS DE MAJUSCULE !)
Quels sont vos nom et prénom ?
      [Unknown]:  10.76.211.91
Quel est le nom de votre unité organisationnelle ?
      [Unknown]:  Caisse de compensation du Valais
Quel est le nom de votre entreprise ?
      [Unknown]:  CCCVs
Quel est le nom de votre ville de résidence ?
      [Unknown]:  Sion
Quel est le nom de votre état ou province ?
      [Unknown]:  Valais
Quel est le code pays à deux lettres pour cette unité ?
      [Unknown]:  CH
Est-ce CN=10.76.211.91, OU=Caisse de compensation du Valais, O=CCCVs, L=Sion,
ST=Valais, C=CH ?
      [non]: oui
Entrez le mot de passe de la clé pour <tomcat>
        (appuyez sur Entrée s'il s'agit du mot de passe du fichier de clés) :
[ENTER]
```

17.1.2. MISE EN PLACE DU CONNECTEUR HTTPS

Dé-commenter la zone "*SSL HTTP/1.1 Connector*" du fichier **C:/Program Files/Apache Software Foundation/Tomcat 7.0/conf/server.xml**.

```
<Connector
       port="8443" SSLEnabled="true"
       maxThreads="150" scheme="https" secure="true" maxPostSize="0"
       keystoreFile="C:/files/test.keystore" keystorePass="k*******"
       clientAuth="false" sslProtocol="TLS" disableUploadTimeout="true" />
```

17.1.3. FORCER LA NAVIGATION EN HTTPS

À ce niveau, si on redémarre le serveur Tomcat, lorsqu'on essaye d'atteindre l'application, on peut le faire en HTTPS sur le port 8443 ET en HTTP sur le port 8080. Mais ce n'est pas ce que je veux. Je souhaite forcer la connexion sécurisée en HTTPS.

Pour ce faire, ajouter les lignes suivantes dans le fichier **/conf/web.xml.** De cette manière, on force la connexion HTTPS côté serveur (et non pas coté client avec une balise meta de redirection en HTML du type `<meta http-equiv="refresh" content="0; URL=…" />` !)

```
<security-constraint>
    <web-resource-collection>
          <web-resource-name>Protected Context</web-resource-name>
          <url-pattern>/*</url-pattern>
    </web-resource-collection>

    <!-- auth-constraint goes here if you requre authentication -->
    <user-data-constraint>
          <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

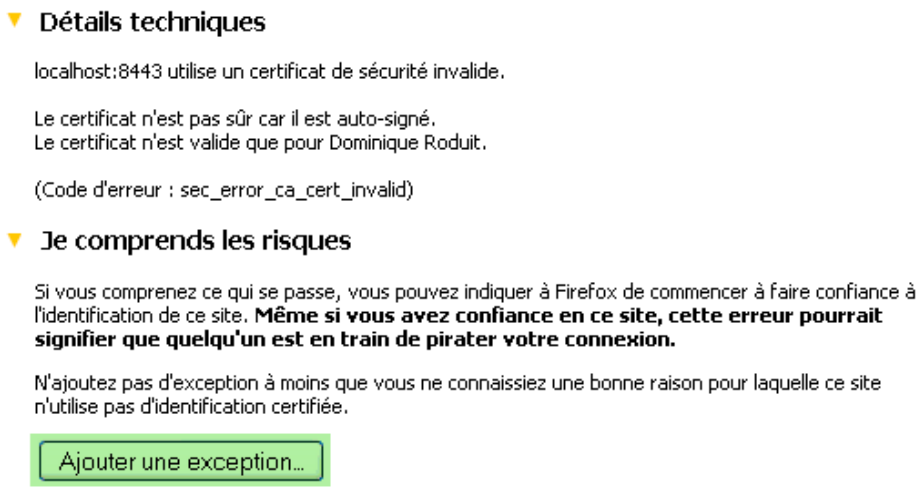
Une fois le fichier édité, il convient de redémarrer le serveur Tomcat.

ⓘ Sur le poste de développement, dans Eclipse, il ne suffit pas de redémarrer le serveur. Il faut le supprimer de la liste des serveurs et le ré-ajouter pour qu'il relance une publication.

Il suffit maintenant de lancer l'application à l'adresse :

```
http://localhost:8443/CCCVsTransfert
```

Malheureusement, comme mon certificat est auto-signé, lors de la première connexion il indique que celle-ci n'est pas certifiée et je dois ajouter une exception pour que l'application soit atteignable.



Une fois l'exception ajoutée, je peux vérifier les informations de mon certificat. Sur Firefox, en allant dans le "Gestionnaire de certificats" du navigateur puis dans "Serveurs".

Internet explorer 8 m'indique une erreur de certificat et colore en rouge la barre d'adresse. Il m'informe que le certificat présenté a été émis pour une autre adresse de site web, cette alerte est compréhensible, car normalement lors de la génération du keystore - quand l'utilitaire demande le Nom/Prénom - c'est en fait le CN (Common Name) qu'il demande. Le CN doit correspondre au nom de domaine mais dans mon cas, je suis encore en local. Durant le développement, je me contenterai donc de naviguer en HTTPS avec cette erreur « normal ».

17.2. SUR LE SERVEUR LINUX

Les commandes à exécuter sont exactement les même qu'en locale sur le poste de développement Windows, à quelques exceptions près.

17.2.1. GÉNÉRATION DU KEYSTORE

Création d'un nouveau dossier nommé "ssl_key" dans "`/opt/tomcat7/`".

```
mkdir /opt/tomcat7/ssl_key/
```

Le keystore généré sera stocké dans ce dossier, il portera le nom "key.keystore".

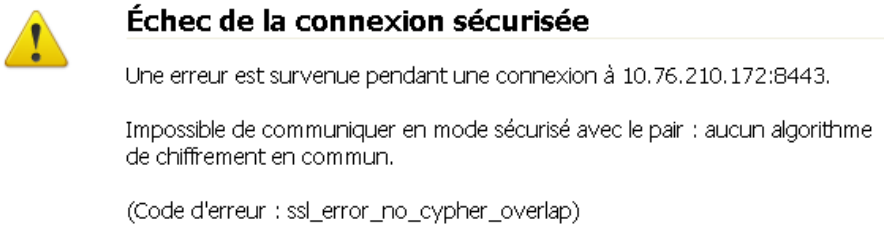
Cette fois-ci, j'indique le nom ou l'adresse IP du serveur comme CN (Nom/Prénom), sinon, il affichera l'erreur comme quoi le certificat a été émis pour une autre adresse de site web.

```
prdWTransfert:/usr/lib64/jvm/jre/bin # keytool -genkey -alias tomcat -keyalg
RSA -keystore /opt/tomcat7/ssl_key/key.keystore
Entrez le mot de passe du fichier de clÃ©s :
Confirmez la saisie du nouveau mot de passe :
Quels sont vos prÃ©nom et nom ?
  [Inconnu(e)] :  10.76.210.172
Comme s'appelle votre unitÃ© d'organisation ?
  [Inconnu(e)] :  Caisse de compensation du valais
Comment s'appelle votre organisation ?
```

```
  [Inconnu(e)] :  CCCVs
Comment s'appelle votre localitÃ© ?
  [Inconnu(e)] :  Sion
Comment s'appelle votre Etat ou votre province ?
  [Inconnu(e)] :  Valais
Quelles sont les deux lettres du code pays pour cette unitÃ© ?
  [Inconnu(e)] :  CH
Le nom CN=10.76.210.172, OU=Caisse de compensation du valais, O=CCCVs, L=Sion,
ST=Valais, C=CH est-il correct ? (taper "oui" ou "non")
  [non] :  oui
```

17.2.2. MISE EN PLACE DU CONNECTEUR HTTPS

Il suffit de suivre exactement les mêmes instructions que pour Windows à une exception près de nouveau. Si le paramètre sslProtocol="TLS", avec internet explorer, je peux naviguer en HTPPS après avoir accepté de poursuivre, sauf qu'avec Firefox, j'ai l'erreur suivante :



En l'état actuel, je ne peux donc pas accéder à mon application. Pour y remédier, il suffit de modifier le paramètre sslProtocol="SSL".

17.2.3. TEST DE CONNEXION

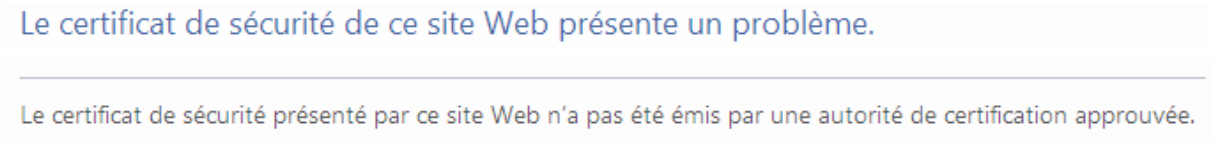
Si j'essaye maintenant d'accéder au serveur avec l'adresse `https://10.76.210.172:8443/` …

17.2.3.1. AVEC FIREFOX

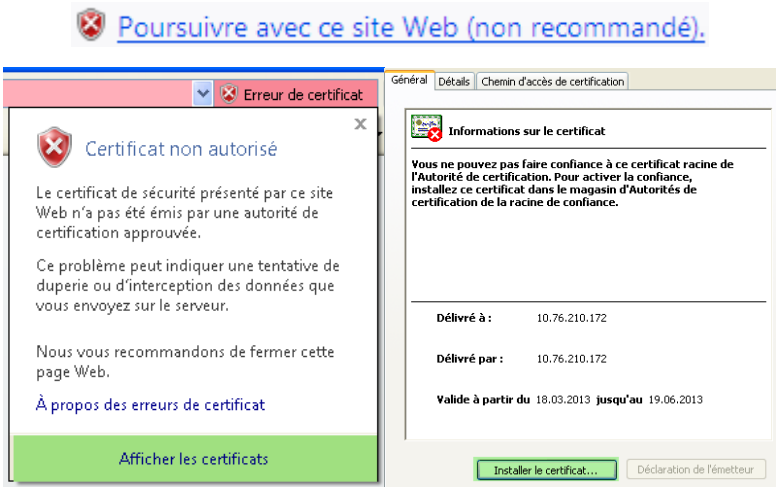
Je dois toujours confirmer l'exception de sécurité lors de la première connexion avec le message : *Le certificat n'est pas sûr car il est auto-signé*.

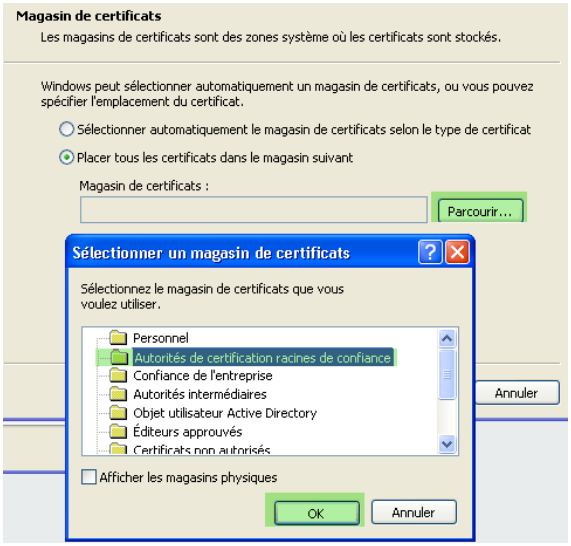
17.2.3.2. AVEC INTERNET EXPLORER 8

J'ai toujours un message d'erreur, mais celui-ci est presque inévitable tant que je n'aurai pas de certificat validé par une autorité de certification approuvée.



Cependant, ce message n'est pas très accueillant et … je ne vais tout de même pas m'arrêter là. Voici donc une parade pour forcer Internet explorer à considérer le certificat comme valide et ainsi ne plus afficher la barre de navigation en rouge.

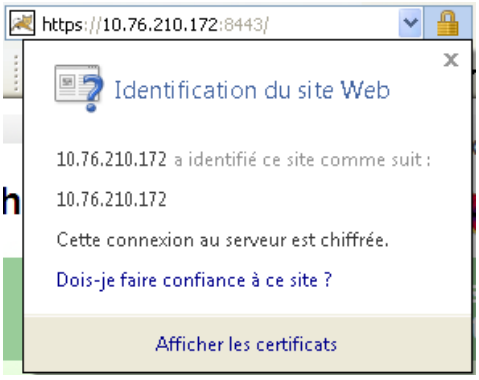




Choisir de placer le certificat dans le magasin "Autorités de certification racines de confiance" pour indiquer à Internet explorer qu'il peut faire confiance au certificat, ainsi, il l'ajoutera dans sa liste blanche.

Redémarrer Internet explorer à la fin de cette opération.

Je n'ai plus de message d'erreur et internet m'indique que la connexion est chiffrée. Le hic est que si j'accède maintenant avec le nom du serveur (https://prdWTransfert:8443/), il me dit que le certificat a été émis pour une autre adresse de site Web. À cause du CN différent…



17.3. VÉRIFICATION DU CHIFFREMENT

Comment savoir ce que fait vraiment le certificat SSL ?

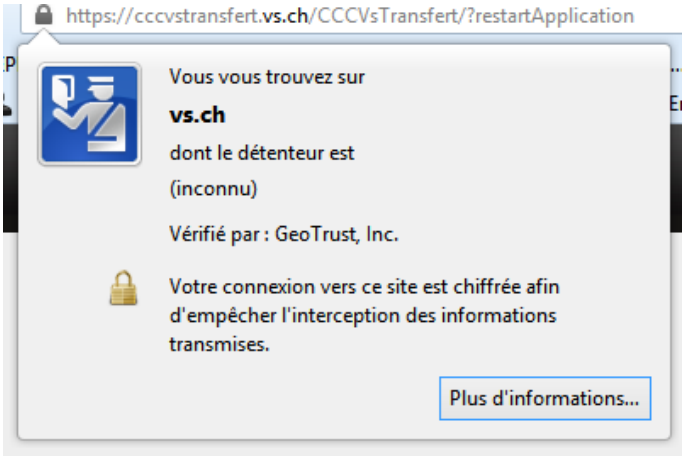
C'est très simple, il suffit d'utiliser un logiciel "sniffer" du type WireShark, qui va permettre d'analyser les paquets qui circulent à travers le réseau. On peut ainsi observer les requêtes envoyées vers le serveur et déterminer si celles-ci sont cryptées ou non.

Je n'ai pas fait ces manipulations car il n'est pas permis de le faire dans l'enceinte du réseau de l'entreprise.

17.4. REMARQUE

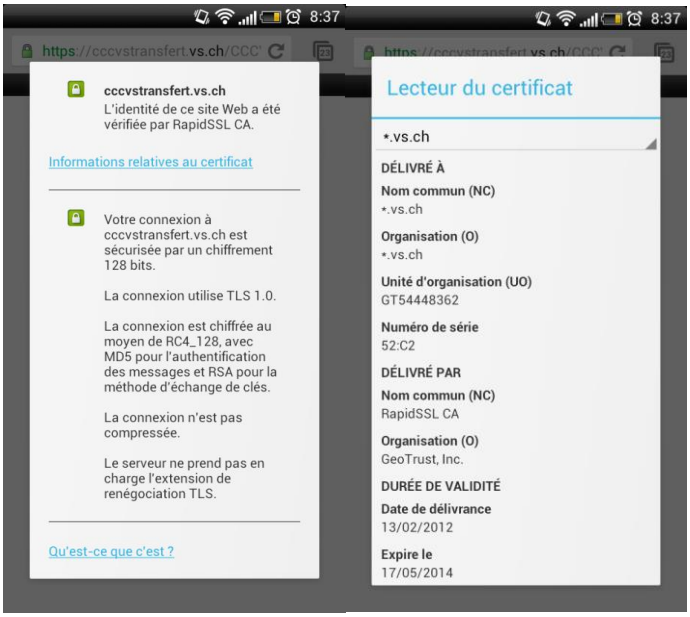
À ma grande surprise, lorsque j'ai accédé à l'application depuis la maison, le jour où la DMZ était configurée (14 mars 2013), j'ai remarqué que la connexion était déjà sécurisée par un certificat SSL et que le protocole HTTPS était utilisé par défaut…

La raison est que le domaine vs.ch utilise un certificat validé par GeoTrust. Notre application, étant placée sur un sous-domaine de vs.ch hérite automatiquement de ce certificat.



Voici les informations que j'ai pu recueillir sur ce certificat :

```
CN = *.vs.ch
OU = Domain Control Validated - RapidSSL(R)
OU = See www.rapidssl.com/resources/cps (c)12
OU = GT54448362
O = *.vs.ch
C = CH
```



L'utilisation d'un certificat SSL était une tâche de mon cahier des charges. C'est pour cela que, malgré l'existence du certificat de vs.ch (que nous n'avions pas prévu), j'ai tout de même effectué tout le travail, toutes les recherches et toute la documentation à ce sujet, de cette manière, je m'assure que le cahier des charges soit strictement respecté.

18. GÉNÉRATION JAVADOC

Pour générer la JavaDoc, j'ai utilisé un outil java tout simple mais très efficace. C'est un programme .jar nommé "aurigadoclet" téléchargé sur le Web. Il suffit de décompresser le contenu du .jar dans un dossier à la racine du disque C. Par exemple C:\doclet.

Dans le dossier "src" du projet Java pour lequel on souhaite générer la Javadoc, il faut créer un fichier de commande MS-DOS (.bat).

Et voici les instructions du fichier .bat :

```
"C:\Program Files\IBM\SDP\jdk\bin\javadoc.exe" -doclet
com.aurigalogic.doclet.core.Doclet -docletpath
"C:\doclet\aurigadoclet\bin\AurigaDoclet.jar" -format pdf -out CCCVs-
Transfert.javadoc.pdf main toolbox web.body
Pause
```

Vous l'aurez compris, il faut renseigner le chemin d'accès au fichier AurigaDoclet.jar et javadoc.exe qui est installé automatiquement avec le JDK (Java development kit), ou dans mon cas, avec Rational Business Developper.

-out : Le chemin vers le fichier de sortie suivi des différents packages pour lesquels on souhaite générer la JavaDoc.

Pour générer la JavaDoc, il suffit maintenant d'exécuter le fichier .bat.

Attention !
Si on utilise des balises HTML non autorisées, par exemple `<img src … />`, l'outil créera un document PDF vide de 0 Octet.

CCCVs-Transfert

Rapport de tests

Caisse de compensation du canton du Valais
Ausgleichskasse Wallis

Service informatique

Dominique Roduit

27/03/2013

dominique.roduit@avs.vs.ch

Base de données

Rapport de test

Description

- Test de la présence de tous les champs nécessaires
- Correspondance avec le schéma entité-relation
- Relations, références et contraintes correctes et existantes

Champs des tables				Correction
trans_users Stockage des informations sur les utilisateurs. Authentification par adresse e-mail et mot de passe.				
PKNoUser	**clé**	Clé primaire de la table	✓	
user_mail		Adresse e-mail de l'utilisateur	✓	
user_pass		Mot de passe crypté en SHA256	✓	
user_internal		Utilisateur interne à la CCCVs ou non	✓	
user_validation		Adresse validée ou non	✓	
user_validation_date		Date de validation de l'adresse e-mail	✓	
user_langue		Langue de l'utilisateur	✓	
user_admin		Définit si l'utilisateur est administrateur	✗	Ajouté le 25.03.13
trans_contacts Liste des contacts ajoutés par les utilisateurs				
PKNoContact	**clé**	Clé primaire de la table	✓	
FKNoUser	clé	Clé étrangère. ID Utilisateur	✓	
contact_name		Nom du contact	✓	
contact_mail		Adresse e-mail du contact	✓	
contact_creation_date		Date de création du contact	✓	
contact_internal		Contact interne à la CCCVs ou non	✓	
trans_folders Conteneur des fichiers d'un transfert				
PKNoFolder	**clé**	Clé primaire de la table	✓	
FKNoUser	clé	Clé étrangère. ID Utilisateur	✓	
folder_name		Nom du dossier de transfert	✓	
folder_description		Description du dossier	✗	Ajouté le 15.03.13
folder_creation_date		Date de création du dossier	✓	
folder_expiration		Date de destruction du dossier	✓	
folder_archive		Dossier archivé (détruit) ou valable	✓	
trans_recipients Liste des destinataires pour un transfert				
PKNoRecipient	**clé**	Clé primaire de la table	✓	
FKNoFolder	clé	Lien vers le dossier de transfert	✓	
FKNoContact	clé	Lien vers le destinataire du transfert	✓	
trans_files Fichiers d'un transfert				
PKNoFile	**clé**	Clé primaire de la table	✓	
FKNoFolder	clé	Assignation d'un fichier à un dossier	✓	
file_name		Nom du fichier (sans le chemin)	✓	
file_rename		Nom du fichier spécifié par l'utilisateur	✓	
file_size		Taille du fichier (en octets)	✓	
file_extension		Extension du fichier	✓	

file_description		Description du fichier	✓	
trans_journal_folders Journalisation des téléchargements et consultations des invités aux dossiers				
PKNoJourFolder	**clé**	Clé primaire de la table	✓	
FKNoContact	clé	Liaison vers le contact qui a téléchargé	✓	
FKNoFolder	clé	Liaison vers le dossier contenant le fichier	✓	
FKNoFile	clé	Liaison vers le fichier téléchargé	✓	
joufo_action		Action (téléchargement ou consultation)	✗	Ajouté le 14.03.13
joudo_date		Adresse validée ou non	✓	
trans_journal_connections Journalisation de la connexion des visiteurs/utilisateurs.				
PKNoJourConnection	**clé**	Clé primaire de la table	✓	
joco_mail		Adresse e-mail de l'utilisateur	✓	
joco_date		Date de la connexion	✓	
joco_ip		Adresse IP publique de l'utilisateur	✓	
joco_os		Système d'exploitation de l'utilisateur	✓	
joco_browser		Navigateur de l'utilisateur	✓	
joco_action		Action de l'utilisateur ou erreur générée	✓	
joco_comment		Détails sur l'action	✓	
trans_app_param Contient les paramètres de l'application				
PKNoParam	**clé**	Clé primaire de la table	✓	
parm_key		Clé/Nom du paramètre	✓	
parm_value		Valeur du paramètre	✓	
parm_description		Description du paramètre	✓	
trans_translate Contient les textes pour la traduction de l'application				
PKNoTranslate	**clé**	Clé primaire de la table	✓	
trans_label		Mot clé d'une traduction	✓	
trans_fr		Texte en français	✓	
trans_de		Texte en allemand	✓	

Relations		Correction
trans_contacts.FKNoUser → trans_users.PKNoUser Relation entre l'utilisateur authentifié et le contact qu'il a ajouté		
Un utilisateur possède aucun, un ou plusieurs contacts Un contact est attribué à un et un seul utilisateur	✓ ✓	
trans_folders.FKNoUser → trans_users.PKNoUser Relation entre l'utilisateur authentifié et le dossier de transfert qu'il a créé		
Un utilisateur possède aucun, un ou plusieurs dossiers Un dossier est attribué à un et un seul utilisateur	✓ ✓	
trans_recipients.FKNoFolder → trans_folders.PKNoFolder Relation entre le dossier créé de transfert et le contact, destinataire du transfert		
Un dossier possède un ou plusieurs destinataires Un destinataire est attribué à un et un seul dossier	✓ ✓	

trans_recipients.FKNoContact → trans_contacts.PKNoContact Relation pour rendre un contact de la liste destinataire d'un transfert		
Un contact est assigné aucune, une ou plusieurs fois comme destinataire Un destinataire est assimilé à un et un seul contact	✓ ✓	
trans_files.FKNoFolder → trans_folders.PKNoFolder Relation entre le fichier et le dossier qui le contient		
Un fichier est attribué à un et un seul dossier Un dossier possède un ou plusieurs fichiers	✓ ✗	Avant : aucun, un ou plusieurs. Corrigé
trans_journal_download.FKNoContact → trans_contacts.PKNoContact Relation pour identifier quel est le contact qui a téléchargé le fichier		
Un contact est attribué à aucun, un ou plusieurs téléchargements Un téléchargement est effectué par un et un seul contact à la fois	✓ ✓	
trans_journal_folders.FKNoFolder → trans_folders.PKNoFolder Relation pour identifier dans quel dossier le fichier a été téléchargé		
Un dossier est attribué à aucun, un ou plusieurs téléchargements Un téléchargement est effectué sur un et un seul dossier	✓ ✓	
trans_journal_folders.FKNoFile → trans_files.PKNoFile Relation pour déterminer quel fichier a été téléchargé.		
Un fichier est attribué à aucun, un ou plusieurs téléchargements Un téléchargement est effectué pour un et un seul fichier à la fois	✓ ✓	

Contraintes			Correction
trans_contacts.FKNoUser → trans_users.PKNoUser Relation entre l'utilisateur authentifié et le contact qu'il a ajouté			
Modification de l'utilisateur Suppression de l'utilisateur	NO ACTION CASCADE	✓ ✗	Avant : NO ACTION
trans_folders.FKNoUser → trans_users.PKNoUser Relation entre l'utilisateur authentifié et le dossier de transfert qu'il a créé			
Modification de l'utilisateur Suppression de l'utilisateur	NO ACTION CASCADE	✓ ✗	Avant : NO ACTION
trans_recipients.FKNoFolder → trans_folders.PKNoFolder Relation entre le dossier créé de transfert et le contact, destinataire du transfert			
Modification du dossier Suppression du dossier	NO ACTION CASCADE	✓ ✗	Avant : NO ACTION
trans_recipients.FKNoContact → trans_contacts.PKNoContact Relation pour nommer un contact en tant que destinataire d'un transfert			
Modification du contact Suppression du contact	NO ACTION CASCADE	✓ ✗	Avant : NO ACTION

trans_files.FKNoFolder → trans_folders.PKNoFolder Relation entre le fichier et le dossier qui le contient			
Modification du dossier Suppression du dossier	NO ACTION CASCADE	✓ ✗	Avant : NO ACTION
trans_journal_download.FKNoContact → trans_contacts.PKNoContact Relation pour identifier quel est le contact qui a téléchargé le fichier			
Modification du contact Suppression du contact	NO ACTION CASCADE	✓ ✗	Avant : NO ACTION
trans_journal_folders.FKNoFolder → trans_folders.PKNoFolder Relation pour identifier dans quel dossier le fichier a été téléchargé			
Modification du dossier Suppression du dossier	NO ACTION CASCADE	✓ ✗	Avant : NO ACTION
trans_journal_folders.FKNoFile → trans_files.PKNoFile Relation pour déterminer quel fichier a été téléchargé.			
Modification du fichier Suppression du fichier	NO ACTION CASCADE	✓ ✗	Avant : NO ACTION

Interface web

Rapport de test

Description

- Navigation : correspondance de la navigation avec le schéma conçu au départ
- Affichage des données : correspondantes et correctes
- Ajout de données : ajout possible, gestion des exceptions
- Edition des données : édition possible, gestion des exceptions
- Suppression des données : suppression possible, gestion de l'intégrité des données

Navigation		Correction
Accès Lors de l'accès à l'application CCCVs-Transfert		
Nous arrivons sur une page de connexion	✓	
Il n'est pas possible d'interagir sans se connecter	✓	
Lors de la première connexion		
Le ruban affiche "Ajouter un contact"	✓	
Le panneau pour la gestion des contacts est sélectionné	✓	
Si l'utilisateur est un administrateur		
Un onglet "Administration" est disponible	✓	
Connexion Comportement de la page de connexion		
Si aucune information n'est saisie dans le formulaire		
Aucun message d'erreur n'est affiché	✓	
La connexion n'est pas effectuée	✓	
● L'action est journalisée	✓	
Si l'adresse e-mail est saisie dans un format invalide		
Un message d'erreur adapté est affiché	✓	
La connexion n'est pas effectuée	✓	
● L'action est journalisée	✓	
Si la longueur du mot de passe saisi est < 3 ou > 50		
Un message d'erreur adapté est affiché	✓	
La connexion n'est pas effectuée	✓	
Si l'adresse e-mail saisie n'existe pas dans la base de données		
Un lien de validation est envoyé à l'adresse mentionnée	✓	
Un message est affiché à l'utilisateur pour l'avertir de l'envoi du mail	✓	
L'utilisateur est enregistré dans la base de données	✓	
● L'action est journalisée	✓	
La connexion n'est pas effectuée	✓	
Si l'adresse e-mail saisie est connue mais n'est pas encore validée		
Un lien de validation est ré-envoyé à l'adresse mentionnée	✓	
Un message est affiché à l'utilisateur pour l'informer sur la validation	✓	
● L'action est journalisée	✓	
Si l'adresse e-mail est connue, validée mais que le mot de passe est erroné		
Un message d'erreur adapté est affiché	✓	
● L'action est journalisée	✓	
La connexion n'est pas effectuée	✓	

Si l'adresse e-mail saisie est connue, validée et que le mot de passe est bon		
L'utilisateur accède à l'interface	✓	
● L'action est journalisée	✓	
Si une exception SQL est déclenchée		
Un message d'erreur adapté est affiché	✓	
● L'action est journalisée	✓	
Si la connexion est réussie		
On accède avant tout à la page d'accueil	✗	Corrigé 07.03.13
Il est possible de naviguer à travers l'application, dans les pages	✓	
Contacts	✓	
Transferts	✗	Ajouté 07.03.13
Conditions d'utilisations	✗	Ajouté 07.03.13
À propos		
Validation des comptes Validation du compte de l'utilisateur à travers le lien crypté qui lui est envoyé par e-mail		
Le lien contenu dans le mail reçu		
est sécurisé par un cryptage du paramètre de l'URL	✓	
redirige vers la page de connexion	✓	
permet de récupérer les informations du bon utilisateur	✓	
valide véritablement le compte de l'utilisateur	✓	
connecte automatiquement l'utilisateur après validation	✗	Corrigé 08.03.13
Si l'utilisateur valide son compte avec succès pour la 1e fois		
Le compte est validé	✓	
● L'action est journalisée	✓	
Si l'utilisateur tente de valider son compte plusieurs fois		
Un message est affiché pour indiquer que l'adresse est déjà validée	✓	
● L'action est journalisée	✓	
Si la valeur du paramètre dans l'URL est modifiée et n'est pas valable		
Aucun message d'erreur n'est affiché	✓	
● L'action est journalisée	✓	
Si des caractères susceptibles de provoquer une injection SQL sont insérés (')		
Les caractères concernés sont considérés comme les autres	✓	
Contacts Page de gestion des contacts de l'utilisateur		
Il est possible d'exécuter les modules :		
Ajouter un contact	✓	
Editer un contact	✓	
Supprimer un contact	✓	
Transferts Page d'affichage et de création des dossiers de transferts de l'utilisateur		
Il est possible d'exécuter les modules		
Nouveau dossier	✓	
Etape 1 – Création du dossier	✓	
Etape 2 – Ajout des destinataires	✓	
Etape 3 – Ajout des fichiers	✓	
Etape 4 – Renommage des fichiers transférés	✗	Ajouté 08.03.13

Il est possible de consulter		
Les informations de chaque dossier séparément	✗	Ajouté 07.03.13
Les actions journalisées pour les utilisateurs d'un dossier	✗	Ajouté 18.03.13
La fenêtre de récupération des liens de fichiers	✗	Ajouté 21.03.13
La fenêtre de récupération des liens des archives des dossiers	✗	Ajouté 21.03.13
La fenêtre pour le renvoi des liens aux contacts inclus au dossier	✗	Ajouté 21.03.13
Téléchargements Page de téléchargement des fichiers d'un dossier		
Il est possible d'accéder à la page téléchargement d'un dossier en suivant le lien reçu par mail	✗	Ajouté 08.03.13
La page reconnait l'utilisateur qui accède depuis les paramètres URL cryptés	✓	
L'accès est refusé si		
un des paramètres crypté est modifié	✓	
le contact est supprimé de la liste des destinataires	✓	
le dossier est archivé (expiré)	✓	
La page se traduit en fonction de la langue du navigateur du client	✓	
Il est possible de télécharger un fichier via le bouton "Télécharger"	✓	
Il est possible de télécharger un fichier via le menu contextuel	✓	
Il n'est plus possible d'accéder à un dossier dont la durée de vie a expiré.	✗	Corrigé 14.03.13
Un bouton "Télécharger tous" permet de télécharger une archive au format ZIP, contenant tous les fichiers du dossier	✗	Ajouté 11.03.13
Conditions d'utilisation Page affichant les conditions d'utilisation de l'application		
Il est possible de revenir au dernier module sélectionné	✗	Ajouté 08.03.13
Un bouton "Manuel utilisateur" affiche le manuel d'utilisation au format PDF	✗	Ajouté 15.03.13
À propos Page affichant des informations sur l'application et les droits du propriétaire de l'application		
Il est possible de revenir au dernier module sélectionné	✗	Ajouté 08.03.13
Administration Onglet de l'interface utilisateur qui permet aux administrateurs de visualiser et gérer les données		
Page "Utilisateurs"		
L'accès n'est permis qu'aux administrateurs de l'application	✓	
Un menu contextuel permet les actions suivantes :		
Valider/Invalider	✓	
Administrateur/Utilisateur simple	✓	
Supprimer	✓	
Page "Journalisations"		
L'accès n'est permis qu'aux administrateurs de l'application	✓	
Il est possible d'afficher les données de chaque catégorie	✓	
Il est possible de filtrer les données à l'aide des boutons du bas	✓	

Affichage des données		Correction
Liste des dossiers (menu) L'affichage des données de la liste est correct et correspond aux champs de la base de données		
Champ de l'interface correspond à **Champs de la base de données**		
Dossiers — folder_name	✓	
Différence visuelle entre les dossiers archivés et disponibles	✗	Corrigé 07.03.13
Un menu contextuel est affiché au clic droit sur un dossier avec les options :		
Pour les dossiers archivés :		
- Supprimer	✗	Corrigé 07.03.13
Pour les dossiers disponibles :		
- Télécharger	✗	Ajouté 14.03.13
- Récupérer le lien	✗	Ajouté 21.03.13
- Renvoyer le lien…	✗	Ajouté 21.03.13
Création d'un nouveau dossier Assistant pour l'enregistrement d'un nouveau dossier		
Les champs suivants sont affichés et leur validité est vérifiée :		
Nom du dossier (Obligatoire)	✓	
Durée de vie (jours)	✓	
Description	✓	
Ajout des destinataires Assistant pour l'ajout de destinataires aux dossiers		
Liste des contacts du champ de sélection		
Champ de l'interface correspond à **Champs de la base de données**		
Pas de nom — contact_name	✓	
Tableau d'affichage des destinataires sélectionnés		
Champ de l'interface correspond à **Champs de la base de données**		
Nom — contact_name	✓	
Adresse e-mail — contact_mail	✓	
Interne — contact_internal	✓	
Un menu contextuel est affiché au clic droit sur un élément du tableau	✓	
Ce menu permet de supprimer une ligne	✓	
Si aucun destinataire n'est sélectionné, il n'est pas possible de continuer	✓	
Transfert de fichiers Assistant pour le transfert de fichiers		
Champ de l'interface correspond à **Champs de la base de données**		
Nom — file_name	✓	
Taille — file_size	✓	
Progression — -	✓	
Etat — -	✓	

Sur les navigateurs récents, une zone permet de glisser-déposer les fichiers	✓	
Sur les anciens navigateurs, une zone qui demande de sélectionner les fichiers est affichée à la place de la zone de drag&drop.	✓	
Le tableau affiche les fichiers choisis et implémente un menu contextuel	✓	
Les fichiers enregistrés n'ont pas d'accentuations dans leurs noms	✗	Corrigé 14.03.13
Un bouton permet le renommage des fichiers une fois le transfert terminé	✗	Ajouté
Si tous les fichiers sélectionnés sont supprimés du tableau, le bouton terminé et celui qui permet de renommer ne sont plus visibles.	✗	Corrigé
Une barre de progression est affichée pour l'avancement global du transfert	✗	Ajouté
Le bouton "Annuler" qui permet de quitter l'assistant est affiché si tous les fichiers transférés sont supprimés.	✗	Corrigé
Plusieurs fichiers sélectionnés de plus de 1Go ne sont pas autorisés.	✗	Corrigé
Il est possible de déterminer la taille maximum pour un transfert (paramètre)	✓	Ajouté
Il est possible de n'autoriser que certaines extensions de fichier (paramètre)	✓	Ajouté
Une zone affiche la taille de tous les fichiers sélectionnés	✗	Ajouté
Liste des contacts dans le menu Tableau contenant la liste des contacts dans le menu		
Champ de l'interface correspond à **Champs de la base de données**		
Nom — contact_name	✓	
Un menu contextuel est affiché au clic droit d'un contact, et permet de : Supprimer ce contact	✓	
Lorsqu'un contact est supprimé, l'action est répliquée dans le tableau du corps de page	✓	
Une icône est affichée à côté des contacts et indique si ceux-ci sont internes ou externes.	✗	Ajouté 18.03.13
Liste des contacts Tableau contenant la liste des contacts de l'utilisateur		
Champ de l'interface correspond à **Champs de la base de données**		
Nom — contact_name	✓	
Adresse e-mail — contact_mail	✓	
Type — contact_internal	✓	
Dernière modification — contact_creation_date	✓	
Un menu contextuel est affiché au clic droit sur un contact, avec les options : Supprimer Editer	✓ ✓	
L'option « Editer » ouvre une fenêtre permettant la mise à jour du contact	✓	
Une icône est affichée à côté des contacts et indique si ceux-ci sont internes ou externes.	✗	Ajouté 18.03.13
La date affichée est correctement formatée	✓	
Lorsqu'un contact est supprimé, l'action est répliquée dans le tableau du corps de page	✓	

Informations sur un dossier sélectionné
Informations affichées lorsqu'un dossier de transfert est sélectionné

Champ de l'interface	correspond à	Champs de la base de données		
Nom du dossier		folder_name	✓	
Date de création		folder_creation_date	✓	
Date d'expiration		folder_expiration	✓	
Nom des destinataires		contact_name	✓	
Nom du fichier		file_name	✓	
Taille du fichier		file_size	✓	
Type de fichier		file_extension	✓	
Une icône illustre le type de fichier en fonction de son extension			✗	Ajouté 18.03.13
Les destinataires sont affichés sous forme de boutons et les actions suivantes sont déclenchées lorsqu'on clique sur l'un d'eux :				
Si des actions sont journalisées pour ce dossier				
Accès à la fenêtre qui affiche les actions journalisées			✗	Ajouté 18.03.13
Si aucune action n'est journalisée pour le dossier				
Un message indique que le contact n'a jamais consulté le dossier			✗	Ajouté 18.03.13
Le temps restant avant la suppression des fichiers et l'archivage est correct			✓	
Les dates de création et d'expiration affichées sont correctes			✓	
Le nombre de fichiers contenus dans le dossier est affiché et correct			✓	
La taille total du dossier est affichée et correcte			✓	
Un indicateur différent permet la distinction des dossiers archivés			✓	
Un menu contextuel est affiché sur le clic droit des fichiers et permet :				
Le téléchargement d'un fichier			✗	Ajouté 18.03.13
La récupération des liens de téléchargement des fichiers			✗	Ajouté 18.03.13

Page de téléchargement
Tableau contenant les fichiers transférés par un utilisateur et disponibles en téléchargement

Champ de l'interface	correspond à	Champs de la base de données		
Nom du dossier		folder_name	✓	
Date de création		folder_creation_date	✓	
Date d'expiration		folder_expiration	✓	
Nom des destinataires		contact_name	✓	
Nom du fichier		file_name	✓	
Taille du fichier		file_size	✓	
Type de fichier		file_extension	✓	
Description du fichier		file_description	✗	Ajouté 11.03.13
Les fichiers peuvent être téléchargés, les images et fichiers textes aussi			✗	Ajouté 18.03.13

Général Sur l'application en général		
Tous Les textes sont traduits dans la langue de l'utilisateur (menus y compris)	×	Corrigé 08.03.13
L'utilisateur peut changer la langue de l'application	×	Corrigé 08.03.13
Le contenu de l'e-mail pour la validation est correct	×	Corrigé 11.03.13
Le contenu de l'e-mail pour le téléchargement est correct	×	Corrigé 11.03.13
Chaque nom de fichier est unique et n'écrase pas un fichier plus ancien	×	Corrigé 11.03.13
Les textes des mails sont affichés correctement, sans problème d'encodage	×	Corrigé 11.03.13
Les dates sont affichées correctement, dans un bon format	×	Corrigé 15.03.13
Envoi d'e-mails E-mails envoyés par l'application aux utilisateurs		
À la fin d'un transfert, sur le clic du bouton "Terminer" des e-mails sont envoyés à chacun des destinataires sélectionnés.	✓	
Suppression automatique des fichiers		
Un e-mail de rappel est envoyé lorsque le dossier est en fin de vie.	×	Ajouté 18.03.13
A l'auteur : Inclus un historique résumé et détaillé des journalisations	×	Ajouté 18.03.13
Aux destinataires : Inclus un historique des actions personnelles	×	Ajouté 18.03.13
Un e-mail est envoyé lorsque le dossier est en fin de vie	×	Ajouté 18.03.13
A l'auteur : Inclus un historique résumé et détaillé des journalisations	×	Ajouté 18.03.13
Aux destinataires : Inclus un historique des actions personnelles	×	Ajouté 18.03.13
L'encodage des e-mails est correct, les accents s'affichent correctement	×	Corrigé 25.03.13
Fenêtre de récupération d'un lien de téléchargement Cette fenêtre est disponible via le menu contextuel des dossiers et fichiers de l'interface utilisateur		
Dès l'ouverture de la fenêtre, le lien est sélectionné pour pouvoir être directement copié simplement.	✓	
Il n'est pas possible d'écrire dans la zone qui contient l'URL (Lecture seul)	✓	
Le lien affiché est correct selon l'environnement (développement, production)	✓	
Fenêtre de renvoi du lien d'accès à un dossier aux destinataires Permet de renvoyer le lien de téléchargement d'un dossier aux destinataires sélectionnés		
Si aucun utilisateur n'est sélectionné, un message est affiché	✓	
L'utilisateur peut quitter la fenêtre via un bouton "Annuler"	✓	
L'utilisateur peut sélectionner un ou plusieurs contacts	✓	
Lors du clic sur le bouton "Envoyer", les mails sont réellement envoyés aux destinataires sélectionnés	✓	
Les dates d'expirations indiquées dans le mail sont correctement calculées	✓	
Administration des utilisateurs Page de gestion des utilisateurs de l'interface, uniquement disponible pour les administrateurs		
Il est possible d'assigner un utilisateur simple en administrateur	✓	
Il est possible de ramener un administrateur à son rang de simple utilisateur	✓	

Il est possible de supprimer un utilisateur	✓	
Il est possible de valider le compte d'un utilisateur	✓	
Il est possible d'invalider le compte d'un utilisateur	✓	
Lorsqu'une action est exécutée, la table est mise à jour instantanément.	✓	
Administration des journalisations Page de visualisation des actions journalisées (pas d'édition de données)		
Les catégories suivantes sont présentes et accessibles		
Connexions	✓	
Validation des comptes	✓	
Téléchargements	✓	
Les filtres sont fonctionnels et affichent les résultats attendus		
Connexions Mot de passe faux Connexion réussie Infos invalides Format e-mail invalide Nouvel utilisateur Compte non validé	✓	
Validation des comptes Erreurs Récurrences Valides	✓	
Téléchargements Consultations Archives des dossiers Téléchargements de fichiers	✓	
Lorsqu'un bouton est cliqué, un style visuel indique qu'il est sélectionné	✓	
Les filtres correspondent à la catégorie d'actions sélectionnée	✓	
L'affichage des données est limité à 100 enregistrements par pages.	✓	
La page ne génère pas d'erreur, même en cas de long chargement.	✓	
Les dates, fichiers, adresses e-mail et adresses IP sont correctement formatés	✓	

Ajout de données		Correction
Contacts Formulaire d'ajout d'un contact affiché dans une sous-fenêtre.		
Le nom ne peut pas être vide	✓	
Le nom doit être compris entre 3 et 25 caractères	✓	
L'adresse e-mail ne peut pas être vide	✓	
L'adresse e-mail doit être dans un format correct	✓	
Si l'utilisateur est un externe, il ne peut qu'ajouter des adresses e-mail internes	✓	
L'ajout d'un contact met à jour la liste des contacts et le menu	✓	
Le clic du bouton « Annuler » ferme la fenêtre d'ajout	✓	
Lors de la soumission, le contact ainsi que la liste sont mis à jour	✓	
Dossiers Assistant de création d'un nouveau dossier		
Création du dossier		
Le nom du dossier ne peut pas être vide	✓	
Le nom du dossier doit être compris entre 3 et 50 caractères	✓	
La durée de vie du dossier doit être comprise entre 1 et 10 jours	✓	
La description est facultative	✓	
Le dossier est créé sur l'appui du bouton « Suivant »	✓	
Ajout des destinataires		
Message d'erreur si aucun contact n'est sélectionné comme destinataire	✓	
Impossible de valider la ComboBox avec une valeur vide	✓	
Les contacts sélectionnés sont ajoutés au tableau	✓	
Impossible d'ajouter deux fois le même contact dans le tableau	✓	
Ajout des fichiers		Corrigé 18.03.13
La session de transfert est limitée à 1 Go	✗	Corrigé 14.03.13
Le nom des fichiers est tronqué s'ils sont trop longs	✗	
Dès que les fichiers sont sélectionnés, l'envoi démarre automatiquement	✓	
La progression de l'envoi est affichée en % et dans une barre de progression	✓	
Le bouton « Terminer » n'est activé qu'une fois tous les fichiers téléchargés	✓	
Les fichiers peuvent être ajoutés depuis le bouton ou la zone de drag&drop	✓	
Le tableau n'est affiché que lorsque les 1ers fichiers sont sélectionnés	✓	
Le bouton « Terminer » est initialement invisible	✓	
Le menu contextuel n'est affiché que lorsque le téléchargement est terminé	✓	Ajouté 15.03.13
Le bouton « Renommer les fichiers » n'est affiché qu'à la fin de l'envoi	✗	
Sur le clic du bouton « Terminer »		
Les e-mails sont envoyés aux destinataires	✓	
Les noms des fichiers envoyés sont enregistrés dans la base de données	✓	Corrigé 07.03.13
La liste des dossiers est mise à jour	✗	Ajouté 14.03.13
Une archive contenant tous les fichiers envoyés est créée	✗	

Edition des données		Correction
Contacts Formulaire d'édition d'un contact affiché dans une sous-fenêtre.		
Le nom ne peut pas être vide	✓	
Le nom doit être compris entre 3 et 25 caractères	✓	
L'adresse e-mail ne peut pas être vide	✓	
L'adresse e-mail doit être dans un format correct	✓	
Si l'utilisateur est un externe, il ne peut qu'ajouter des adresses e-mail internes	✓	
L'ajout d'un contact met à jour la liste des contacts et le menu	✓	
Le clic du bouton « Annuler » ferme la fenêtre	✓	
Lors de la soumission, le contact ainsi que la liste sont mis à jour	✓	

Nom et description des fichiers Wizard facultatif pour le renommage et l'assignation d'une description aux fichiers lors de la création d'un dossier de transfert.		
Les noms des fichiers sont modifiables en passant en mode « Renommage »	✗	Ajouté le 15.03.13
Une colonne description est affichée et éditable	✗	Ajouté le 15.03.13
Les nouveaux noms et descriptions ajoutés sont enregistrés correctement	✗	Ajouté le 15.03.13
Si un fichier est supprimé dans le tableau avant de basculer en mode renommage, la modification est prise en compte.	✗	Ajouté le 15.03.13

Suppression des données		Correction
Contacts Liste des contacts.		
Il est possible de supprimer un contact	✓	
Il est possible de supprimer plusieurs contacts simultanément	✓	
Lors de la suppression d'un contact dans une liste, toutes les listes de contacts sont mises à jour.	✓	
Dossiers Liste des dossiers dans le menu latéral		
Il est possible de supprimer un dossier archivé	✗	Corrigé 08.03.13
Il n'est pas possible de supprimer un dossier encore actif	✗	Corrigé 08.03.13
Lors de la suppression d'un dossier archivé		
La page de départ s'affiche	✓	
La liste est mise à jour, les entrées liées au dossier sont effacées	✓	
Assistant d'ajout des destinataires Assistant pour l'ajout des destinataires lors de la création d'un dossier de transfert		
Il est possible de supprimer un contact ajouté au tableau	✓	
Si la création du dossier est interrompue, le dossier n'est pas enregistré	✗	Corrigé 11.03.13
Assistant d'ajout de fichiers Assistant pour l'ajout des fichiers lors de la création d'un dossier de transfert		
Il est possible de supprimer un fichier ajouté au tableau (menu contextuel)	✓	
Lorsqu'on clique sur « Supprimer » le fichier est supprimé physiquement	✗	Corrigé 08.03.13
Si la création du dossier est interrompue, le dossier et les destinataires liés ne sont pas créés	✗	Corrigé 11.03.13
Suppression automatique des fichiers Suppression automatique des dossiers expirés ou invalides		
Les dossiers dont la date a expiré sont archivés	✗	Corrigé 15.03.13
Les fichiers des dossiers dont la date a expiré sont supprimés	✗	Corrigé 15.03.13
Un dossier dont la création a été interrompue par l'utilisateur en rechargeant la page n'est pas enregistré	✗	Corrigé 11.03.13
L'archive ZIP du dossier expiré est supprimée	✗	Corrigé 15.03.13

Test général

Rapport de test

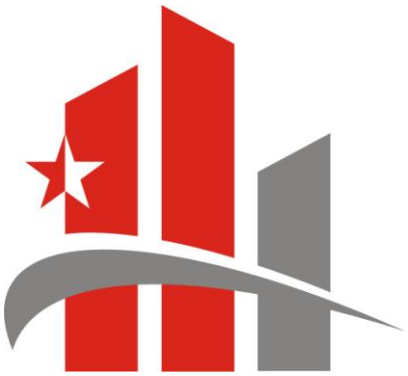
Description

Test général et final de l'application :

- Vérification de la compatibilité et de la performance sur plusieurs navigateurs
- Vérification de la bonne réalisation des objectifs du cahier des charges

Compatibilité multi-navigateurs		Remarque
Internet Explorer Version 8		
Performances	9/10	Ajout d'un style propre à internet pour les boutons en position absolute qui n'étaient pas affichés.
Affichage (graphismes et styles CSS intacts)	7/10	
Il est possible de se connecter	✓	
Il est possible d'accéder à toutes les pages et d'afficher toutes les données	✓	
Le transfert de fichiers se passe correctement	✓	
La langue du navigateur est détectée	✓	
Le manuel utilisateur au format PDF est affiché	✓	
Les fichiers sont téléchargeables	✓	
Mozilla Firefox Version 17		
Performances	9/10	
Affichage (graphismes et styles CSS intacts)	10/10	
Il est possible de se connecter	✓	
Il est possible d'accéder à toutes les pages et d'afficher toutes les données	✓	
Le transfert de fichiers se passe correctement	✓	
La langue du navigateur est détectée	✓	
Le manuel utilisateur au format PDF est affiché	✓	
Les fichiers sont téléchargeables	✓	
Google Chrome Version 25		
Performances	10/10	
Affichage (graphismes et styles CSS intacts)	10/10	
Il est possible de se connecter	✓	
Il est possible d'accéder à toutes les pages et d'afficher toutes les données	✓	
Le transfert de fichiers se passe correctement	✓	
La langue du navigateur est détectée	✓	
Le manuel utilisateur au format PDF est affiché	✓	
Les fichiers sont téléchargeables	✓	

CCCVS



TPI Transfert web

Préparation de l'infrastructure

Donnet-Monay Adrien

Historique des versions
–
Révision du document

Version	Date	Auteur	Modifications
v 1.0	31.01.13	Adrien Donnet-Monay	Création du document de base
v 1.1	27.03.13	Dominique Roduit	Mise en page et correction du contenu

Installation du Serveur SUSE

1 Description

L'objectif de ce document est de relever et de référencer les informations d'installation et les configurations appliquées dans le cadre de la mise en place du serveur « Linux » utilisé par l'application de transfert web. Il comprend notamment des récapitulatifs de paramétrages ou encore les particularités du serveur en question.

2 Installation du système d'exploitation

Selon la documentation, l'un des systèmes agréés est « Linux Suse 11 », c'est sur ce système que sera installée l'application de « Transfert Web ».

2.1 Paramètres d'installation

Langue	Français
Disposition du clavier	Français (Suisse)
Fuseau horaire	Suisse
Mot de passe root	K*******
Nom d'hôte	prdWTransfert
Nom de domaine	cccvs

Enlever coche dans « Modifier le nom d'hôte via DHCP »
Mettre la coche dans « Ecrire le nom d'hôte dans /etc/hosts »

Configuration de la carte réseau

Interface réseau

➔ Bouton Modifier ➔ Onglet Adresse ➔ Adresse IP statique assignée

IP	10.76.210.192
Masque	255.255.254.0
Nom d'hôte	prdWTransfert

➔ Onglet Général ➔ Zone de pare-feu ➔ Zone interne (non protégé) ➔ Onglet Nom d'hôte

DNS	10.76.210.247, 10.76.0.1

➔ Onglet Routage

Passerelle par défaut	10.76.211.254

Proxy

Adresse Proxy	http://proxy.vs.ch:8080

Cocher « Utiliser le même proxy pour tous les protocoles »

Méthode d'authentification de l'utilisateur

Local (/etc/passwd)

Nouvel utilisateur local

Nom complet de l'utilisateur	cccsysa
Nom d'utilisateur	cccsysa
Password	K*******

Configuration du matériel

Laisser les options par défaut

Installation achevée

Mettre la coche sur « Cloner ce système pour AutoYaST»

3 Installations

3.1 JAVA 6

Installation du package ***java-1_6_0-ibm***, via l'interface de configuration « YaST2 » et le programme de « Software Management ».

Automatiquement, le système à également installé les packages :

- ***java-1_6_0-ibm-font***
- ***package-utils***.

3.2 Tomcat 7

- **Version :** 7.0.35
- **Port :** 8080
- **Répertoire d'installation :** /opt/tomcat7/
- Ajout de deux commandes générales (utilisables n'importe où)
  - **Démarrer le serveur :** ***tomcatstart***
  - **Arrêter le serveur :** ***tomcatstop***
- **Taille maximum des requêtes POST :** Illimitée

  (Fichier /conf/server.xml, Paramètre : maxPostSize=0)

3.2.1 Configuration du manager *(pour le déploiement des fichiers WAR)*

- Identifiant : cccsysa
- Mot de passe : cccsysa
- Taille maximum des fichiers WAR au déploiement : relevée de 50MB à 500MB

3.3 MySQL

Version : 5.6.10

3.3.1 Télécharger les packages sous /work/software/mysql

- MySQL-client-5.6.10-1.sles11.x86_64.rpm
- MySQL-client-5.6.10-1.sles11.x86_64.rpm

3.3.2 Installation de mysql

```
cd /work/software/mysql
rpm -iv MySQL-server-5.6.10-1.sles11.x86_64.rpm MySQL-client-5.6.10-1.sles11.x86_64.rpm
...
Preparing packages for installation...
MySQL-client-5.6.10-1.sles11
MySQL-server-5.6.10-1.sles11
mysql                0:off  1:off  2:on   3:on   4:on   5:on   6:off
2013-02-15 13:06:37 0 [Warning] TIMESTAMP with implicit DEFAULT value is deprecated. Please use --explicit_defaults_for_timestamp server option (see documentation for more details).
2013-02-15 13:06:37 938 [Note] InnoDB: The InnoDB memory heap is disabled
2013-02-15 13:06:37 938 [Note] InnoDB: Mutexes and rw_locks use GCC atomic builtins
2013-02-15 13:06:37 938 [Note] InnoDB: Compressed tables use zlib 1.2.3
2013-02-15 13:06:37 938 [Note] InnoDB: CPU does not support crc32 instructions
2013-02-15 13:06:37 938 [Note] InnoDB: Using Linux native AIO
2013-02-15 13:06:37 938 [Note] InnoDB: Initializing buffer pool, size = 128.0M
2013-02-15 13:06:37 938 [Note] InnoDB: Completed initialization of buffer pool
2013-02-15 13:06:37 938 [Note] InnoDB: The first specified data file ./ibdata1 did not exist: a new database to be created!
…

A RANDOM PASSWORD HAS BEEN SET FOR THE MySQL root USER !
You will find that password in '/root/.mysql_secret'.
You must change that password on your first connect,
no other statement but 'SET PASSWORD' will be accepted.
See the manual for the semantics of the 'password expired' flag.
Also, the account for the anonymous user has been removed.
In addition, you can run:
  /usr/bin/mysql_secure_installation
which will also give you the option of removing the test database.
This is strongly recommended for production servers.
See the manual for more instructions.
Please report any problems with the /usr/bin/mysqlbug script!
The latest information about MySQL is available on the web at
  http://www.mysql.com
Support MySQL by buying support/licenses at http://shop.mysql.com
New default config file was created as /usr/my.cnf and
will be used by default by the server when you start it.
You may edit this file to change server settings
```

3.3.3 Vérification des packages installés

- rpm -qa | grep My
- MySQL-server-5.6.10-1.sles11
- MySQL-client-5.6.10-1.sles11

3.3.4 Démarrage de mysql

cd /etc/init.d

./mysql status

./mysql start

./mysql status

3.3.5 Entrée dans mysql

Pour afficher le mot de passe :

- more /root/.mysql_secret
- mysql -p
- Enter password: ...

3.3.6 Modifier le mot de passe de l'utilisateur root

SET PASSWORD = PASSWORD('K*******');

3.3.7 Supprimer la base de données « test »

/usr/bin/mysql_secure_installation

DROP DATABASE test;

CCCVs-Transfert

Manuel d'utilisation

Caisse de compensation du canton du Valais
Ausgleichskasse Wallis

Service informatique

Dominique Roduit

27/03/2013

dominique.roduit@avs.vs.ch

Présentation du produit

CCCVs-Transfert

CCCVs-Transfert est une plateforme de transfert mise à disposition par la Caisse de compensation du Valais pour l'usage exclusif des collaborateurs internes et externes. Cette application remplacera le service en ligne « WebTransfert ».

Inspirée des plateformes de partage actuelles, elle est conçue de manière intuitive pour vous permettre d'effectuer vos transferts simplement, en toute sécurité et avec un suivi des données que vous diffusez.

Il n'est pas autorisé d'utiliser ce service pour le partage de fichiers personnels.

Pour des raisons de sécurité et dans le but de vous garantir le meilleur support possible, nous enregistrons certaines données vous concernant :

- Adresse e-mail
- Adresse IP publique
- Nom, version et langue par défaut du navigateur

Toutes les données relatives aux transferts, y compris la date et l'heure, sont journalisées. Ces données sont analysées afin de détecter tout éventuel abus.

Les deux langues officielles de l'application sont le Français et l'Allemand.

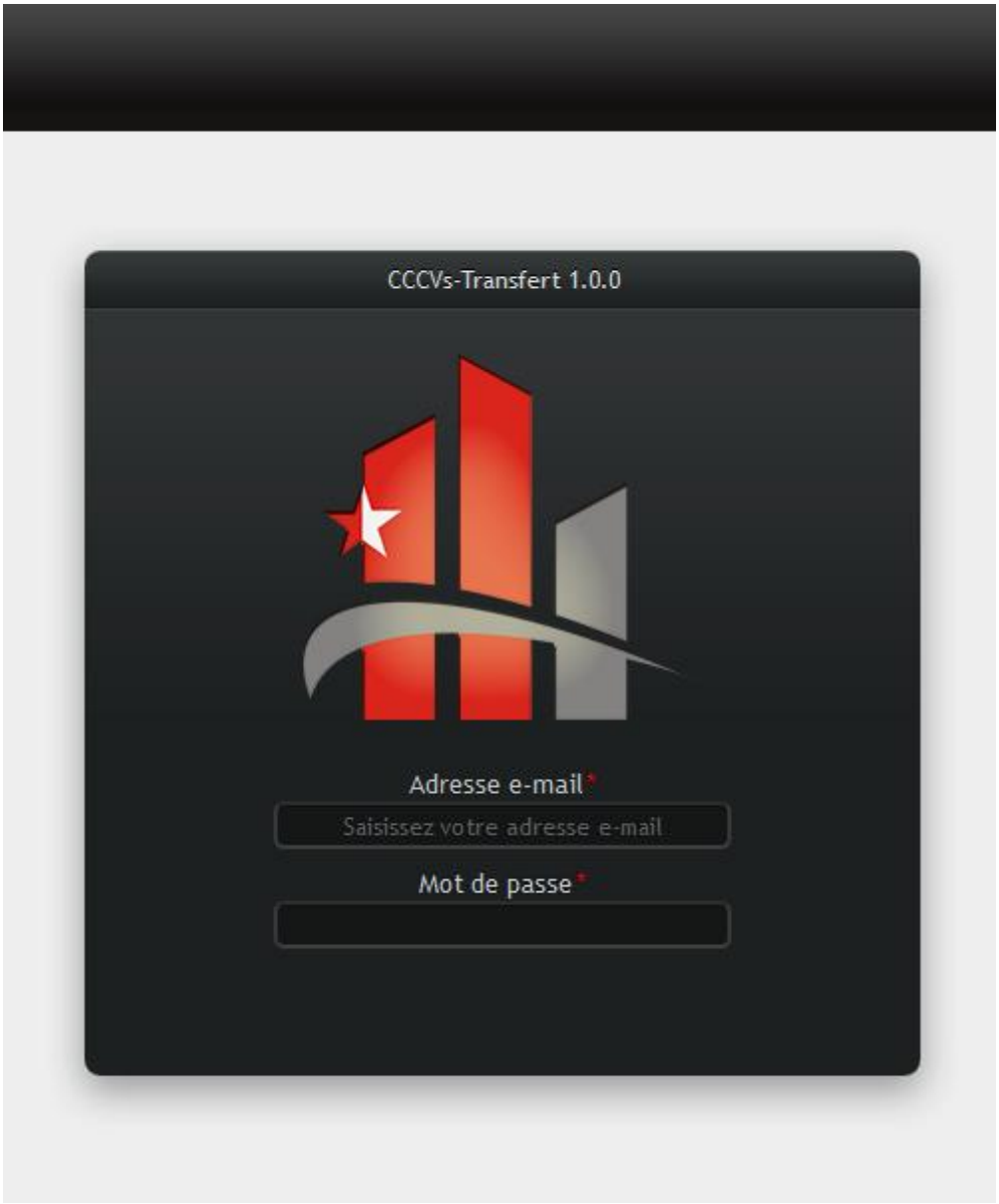
Si vous rencontrez des problèmes techniques, merci de nous les signaler à l'adresse

support@avs.vs.ch

Ce document a pour objectif de vous accompagner lors de votre première prise en main du service.

Inscription

CCCVs-Transfert



Si vous n'êtes pas encore inscrit(e) sur notre service de transfert

1 – Saisissez votre adresse e-mail
2 – Saisissez le mot de passe que vous souhaitez utiliser pour les connexions futures
3 – Validez votre saisie en cliquant sur le bouton « Inscription » qui est apparu

Au terme de ces 3 étapes, un message électronique vous est communiqué. Celui-ci inclut un lien que vous devez valider. Toutes les instructions nécessaires à la validation s'afficheront à l'écran.

Connexion — CCCVs-Transfert

Lors de la validation de votre compte, vous serez automatiquement connecté à nos services.

Lorsque vous souhaitez vous reconnecter à l'application

1 – Rendez-vous à l'adresse https://cccvstransfert.vs.ch/
2 – Saisissez votre adresse e-mail
3 – Saisissez le mot de passe que vous aviez choisi lors de votre inscription
4 – Validez votre saisie en cliquant sur le bouton « Connexion » qui a apparu

Au terme de ces 3 étapes, vous serez connecté à l'interface de transfert.

Modifier la langue de l'interface — CCCVs-Transfert

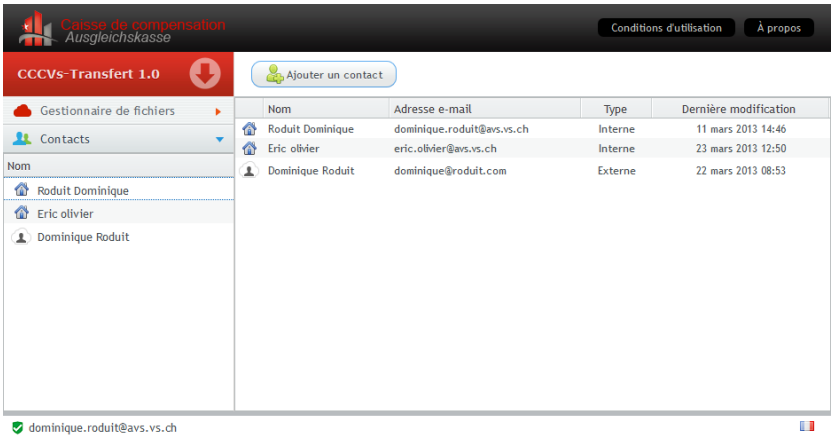
Dans le bord gauche au bas de l'application, vous trouvez une icône de la langue avec laquelle vous utilisez l'application. Pour changer la langue d'affichage, cliquez sur cette icone.

Lors de la première connexion, la langue d'affichage sera la même que celle de votre navigateur.

Ajouter un nouveau contact — CCCVs-Transfert

Avant de pouvoir transférer des fichiers, vous devez ajouter un ou plusieurs contacts. Ces contacts représentent les destinataires de votre futur transfert. Vous pourrez en ajouter/éditer/supprimer à tout moment.

Pour ajouter un contact, rendez-vous dans la rubrique « Contacts » et cliquez sur « Ajouter un contact ». Entrez le nom et l'adresse e-mail de la personne que vous souhaitez ajouter à votre liste.

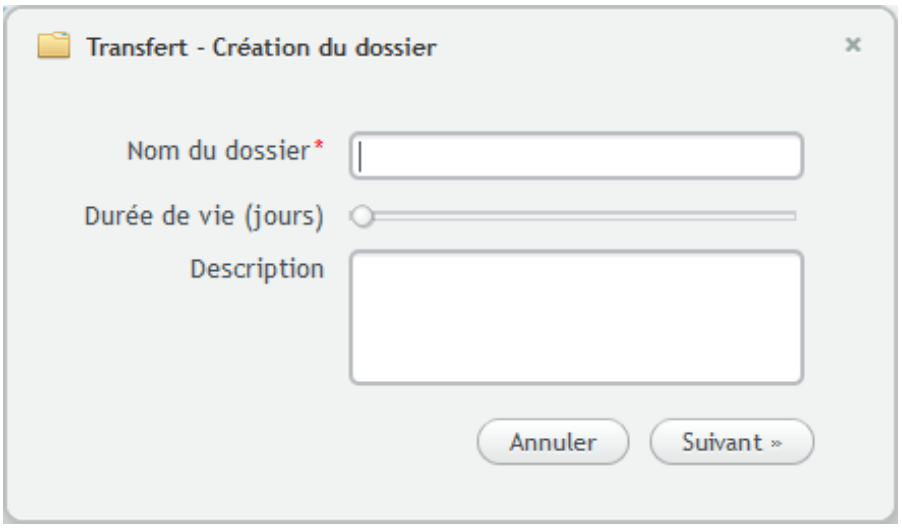


Pour éditer ou supprimer un contact existant, faites un clic droit sur le contact en question puis sélectionnez l'action désirée.

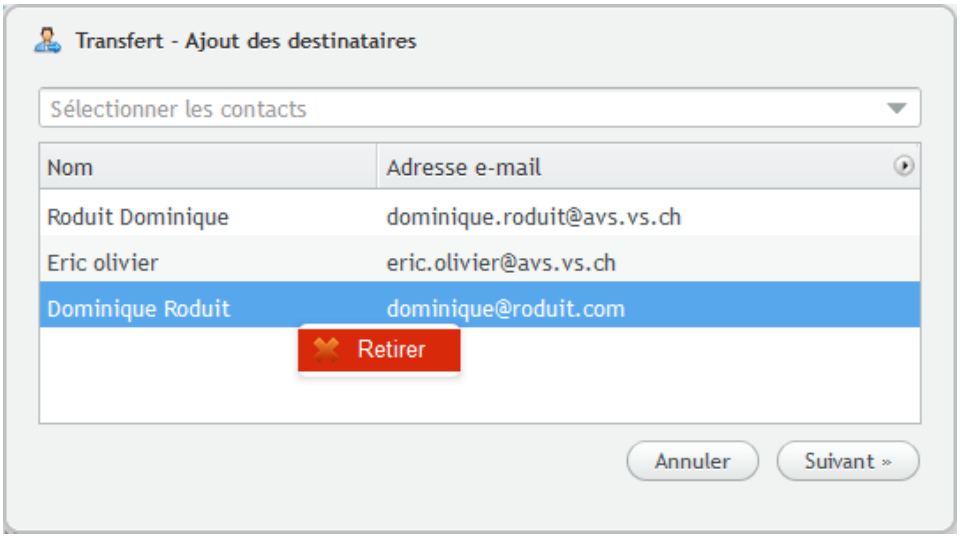
Transférer des fichiers

CCCVs-Transfert

Rendez-vous dans la rubrique « Gestionnaire de fichiers » et cliquez sur « Nouveau dossier ».
Si vous n'avez pas encore ajouté de contact, un message vous demandera d'ajouter les contacts auxquels vous souhaiter envoyer vos fichiers.



1 – Nommer votre dossier *(Vous pouvez créer plusieurs dossiers avec le même nom)*
2 – Indiquez une durée de vie *(Entre 1 et 10 jours)*
La durée de vie correspond au nombre de jours pendant lesquels votre dossier sera consultable par les destinataires auxquels vous partagez vos fichier. Passé la date d'expiration, votre dossier sera archivé et ne sera plus visible par vos contacts.
3 – Entrez une brève description concernant votre transfert de fichier (facultatif).
4 – Cliquez sur « Suivant »

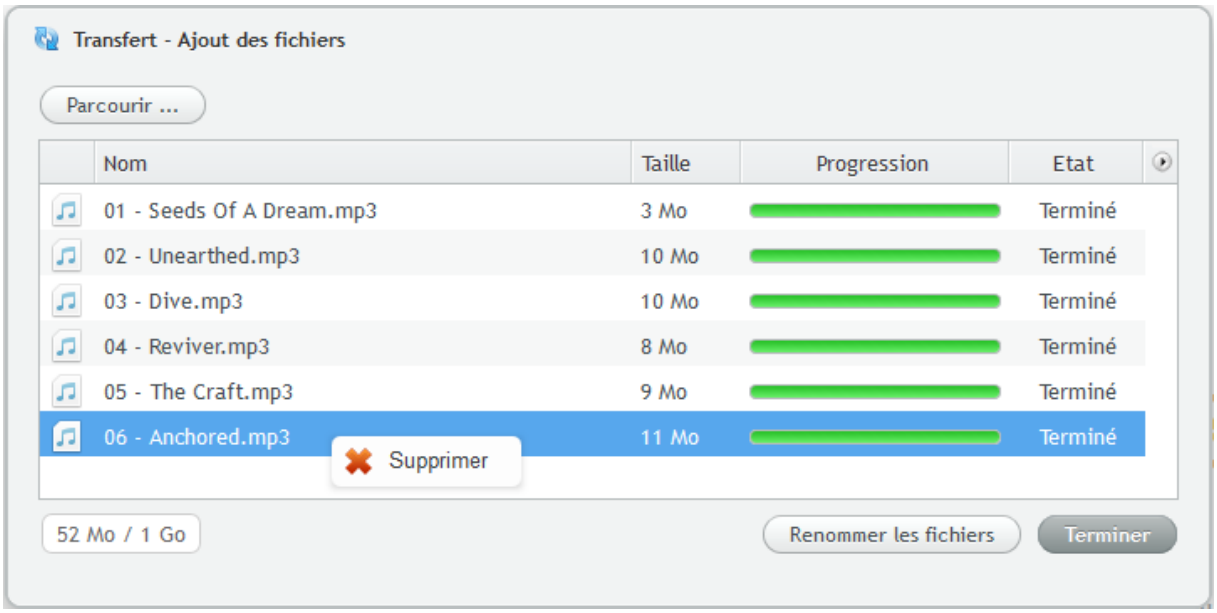


1 – Entrez le nom des destinataires.
Les destinataires sont les contacts de votre liste auxquels vous souhaitez envoyer des fichiers.
Vous pouvez désélectionner des contacts en faisant un clic droit sur la ligne du tableau en question et en cliquant sur « Supprimer ».
2 – Cliquez sur « Suivant »

1 – Sélectionnez les fichiers que vous souhaitez envoyer.

La taille totale des fichiers sélectionnés ne doit pas dépasser 1 Go.

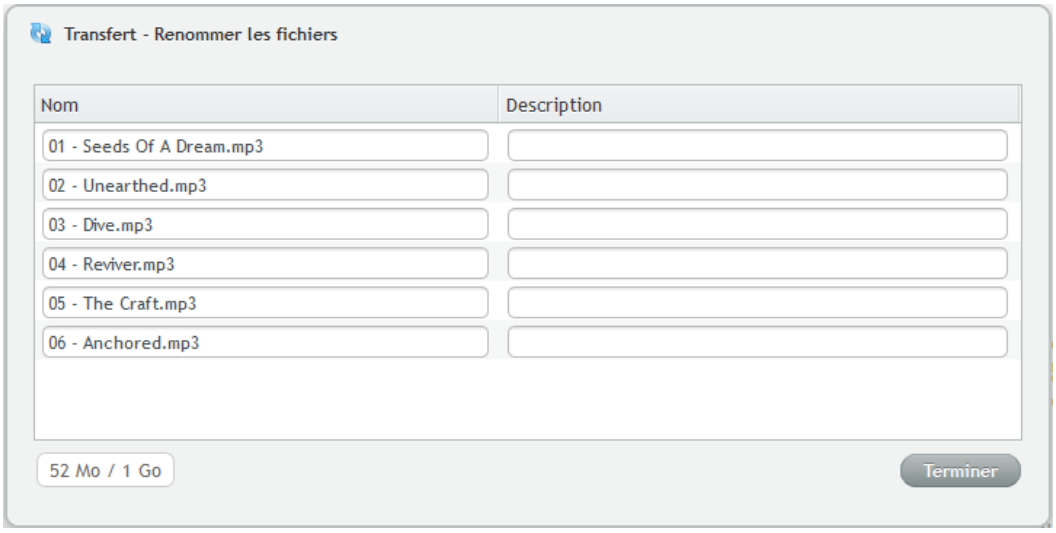
Vous pouvez glisser des fichiers de votre bureau directement dans la fenêtre *(Uniquement avec Firefox, Opéra et Google Chrome)* ou sélectionner vos fichier en cliquant sur « Parcourir… ». Vous pouvez sélectionner plusieurs fichiers à la fois. Vous pouvez envoyer vos fichiers en plusieurs fois et supprimer les fichiers que vous avez déjà envoyés mais que vous souhaitez retirer de la liste en faisant un clic droit sur le fichier concerné puis en sélectionnant l'action « Supprimer ».



2 – Cliquez sur « Renommer les fichiers » ou sur « Terminer » pour envoyer votre dossier tel quel

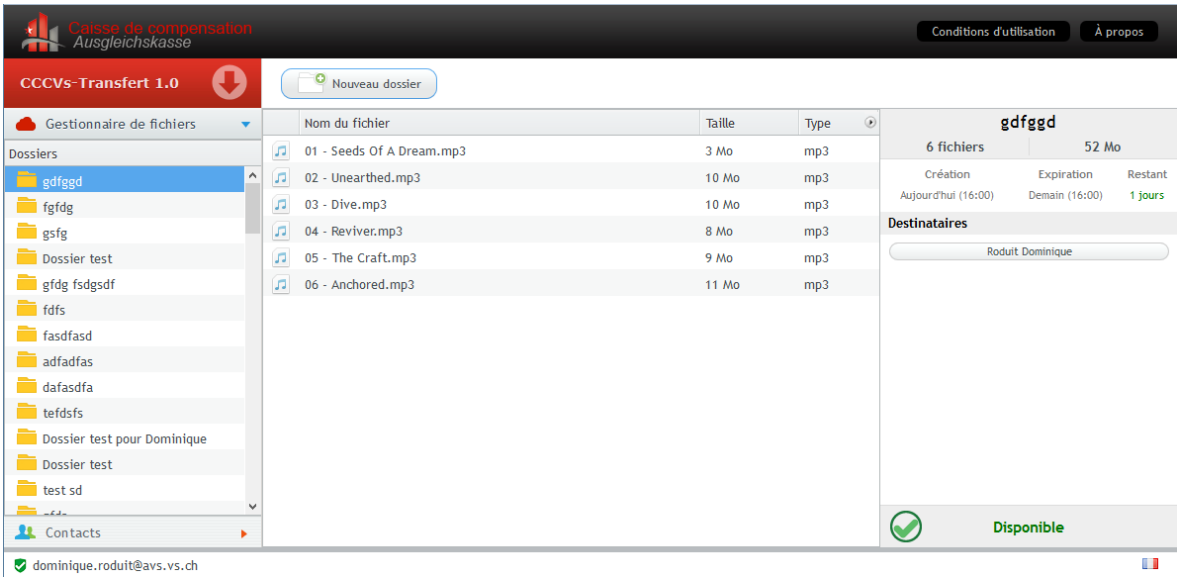
Lorsque vous cliquez sur « Terminer », une archive est générée au format zip, contenant tous les fichiers sélectionnés. Elle permet aux destinataires de télécharger tous les fichiers du dossier en une seule fois. Cette opération peut prendre un peu de temps, laissez donc à l'application le temps d'exécuter ce processus. Le dossier n'est pas créé et les messages électroniques ne sont pas envoyés aux contacts tant que vous ne validez pas votre envoi par le bouton « Terminer ».

Renommer les fichiers

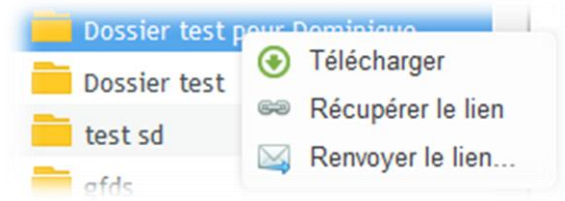


L'assistant « Renommage des fichiers » vous permet de modifier le nom de vos fichiers et de leur ajouter une description. Les destinataires verront alors le nom que vous avez donné à votre fichier au lieu du nom du fichier original (ex. schemas_pile_et electrolyse.pdf).

Résultat



Vous pouvez télécharger l'archive des fichiers de votre dossier en faisant un clic droit sur le dossier en question, dans le tableau du menu latéral gauche qui répertorie la liste de vos dossiers.

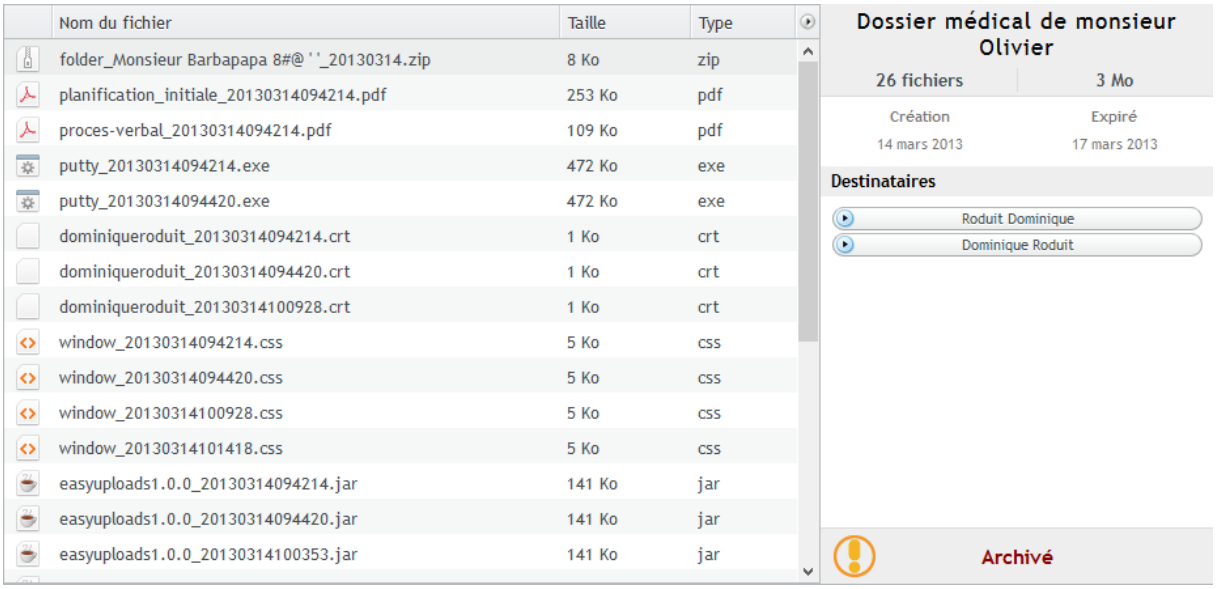


Remarque sur l'expiration des dossiers

Lorsque la durée de vie de votre dossier est dépassée, celui-ci sera archivé automatiquement par le système et tous les fichiers qu'il contient se verront supprimés. Vous pourrez toujours consulter ce dossier dans votre espace personnel mais il ne sera plus téléchargeable et les contacts auxquels vous avez envoyé le dossier n'y auront plus accès. Vous pouvez désormais choisir de supprimer le dossier de la liste en faisant un clic droit sur le dossier en question dans le menu latéral gauche.

Les dossiers archivés sont identifiables par leur icône

Voici comment se présente l'aspect intérieur d'un dossier archivé.

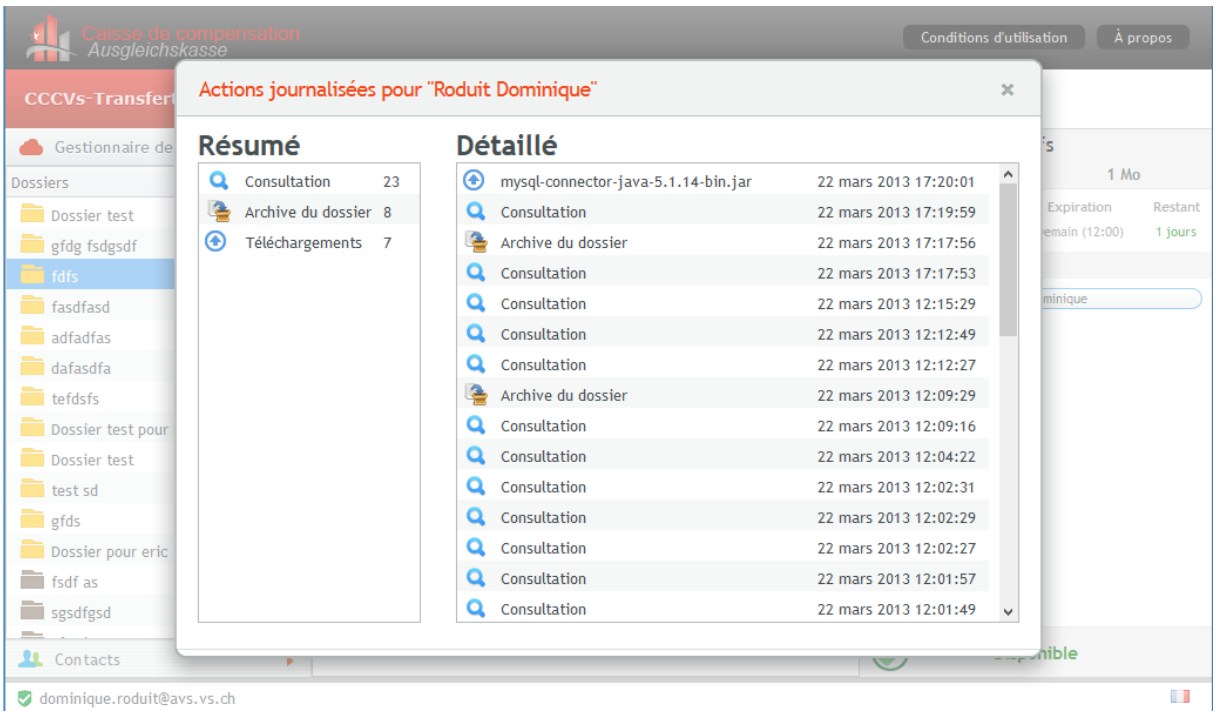


Visualisation de l'historique des journalisations

Lorsque vous avez envoyé un dossier, vous souhaiteriez voir qui a accédé au dossier, quels fichiers ont été téléchargés, à quelle heure… Tout cela est possible. Il suffit de vous rendre dans le dossier pour lequel vous voulez afficher le suivi.

Les destinataires qui ressemblent à cela sont ceux qui ont déjà accédé au dossier :

En cliquant sur la personne désirée, l'historique des actions qu'elle a effectuées et qui ont été journalisées apparaît.

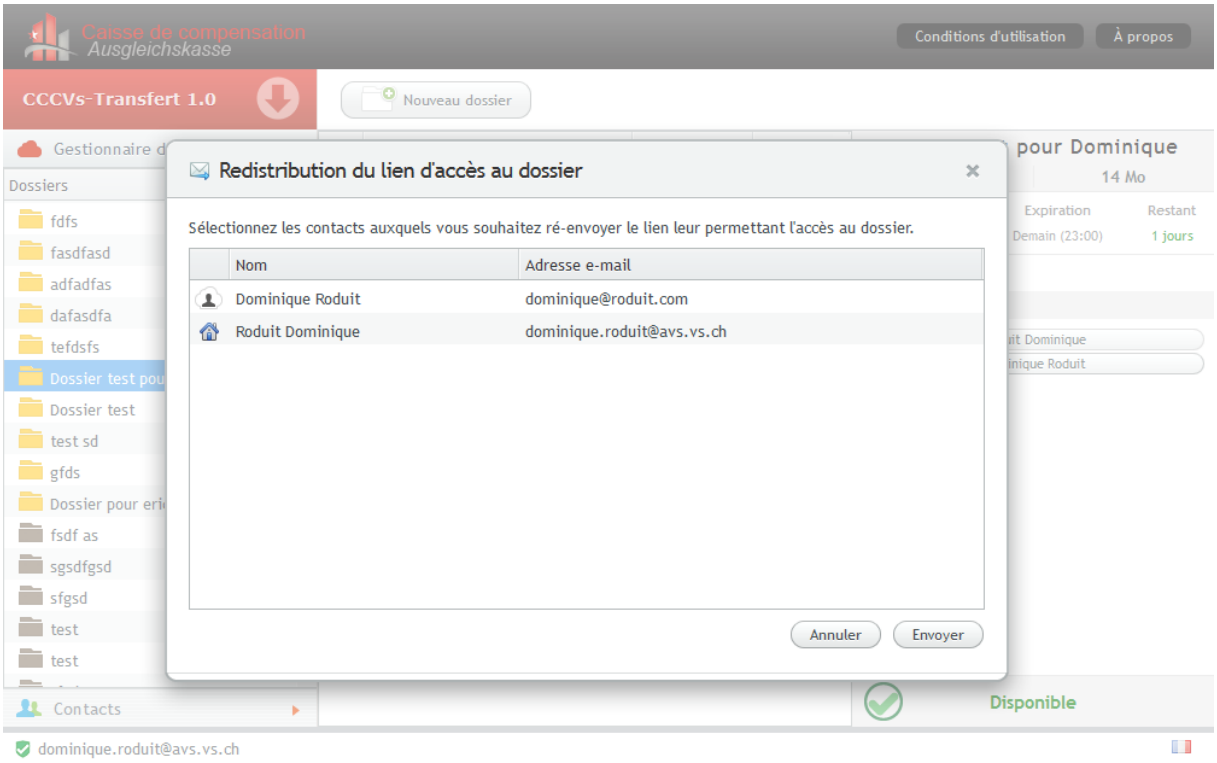


Ceux qui n'ont jamais accédés au dossier n'ont pas l'icône bleue à gauche et lorsque vous cliquez sur eux, le message suivant apparaît :

Renvoi des liens d'accès aux dossiers

Si l'un de vos contacts vous dit qu'il n'a pas reçu le dossier que vous lui avez transmis, vous avez la possibilité de lui renvoyer uniquement le lien d'accès sans devoir renvoyer la totalité des fichiers contenus dans le dossier.

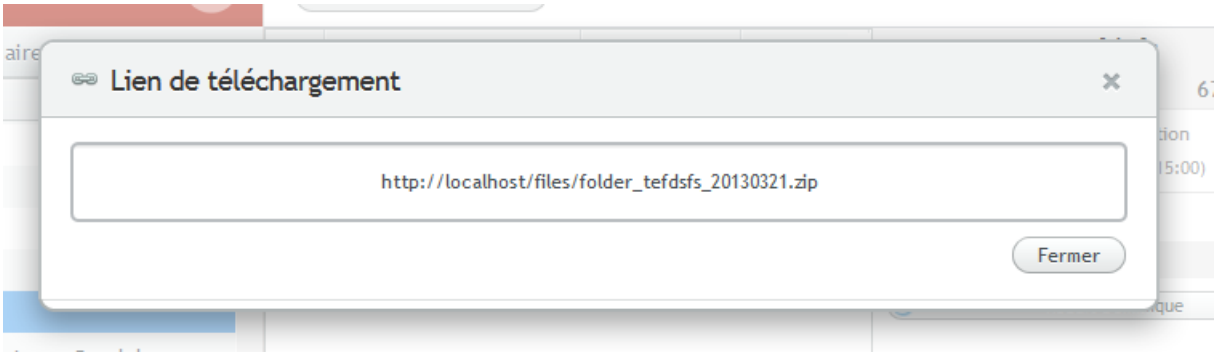
Pour le faire, il suffit de faire un clic droit dans le menu, sur le dossier que vous souhaitez renvoyer, puis de cliquer sur « Renvoyer le lien… ».



Choisissez dans la fenêtre qui s'affiche, les contacts auxquels vous voulez renvoyer le dossier, simplement en cliquant sur eux, puis validez par le bouton « Envoyer ».

Récupération du lien de téléchargement d'un fichier

Vous avez la possibilité de télécharger vos propres fichiers ainsi que de partager le lien d'accès direct au téléchargement de ceux-ci en passant par la fenêtre suivante :



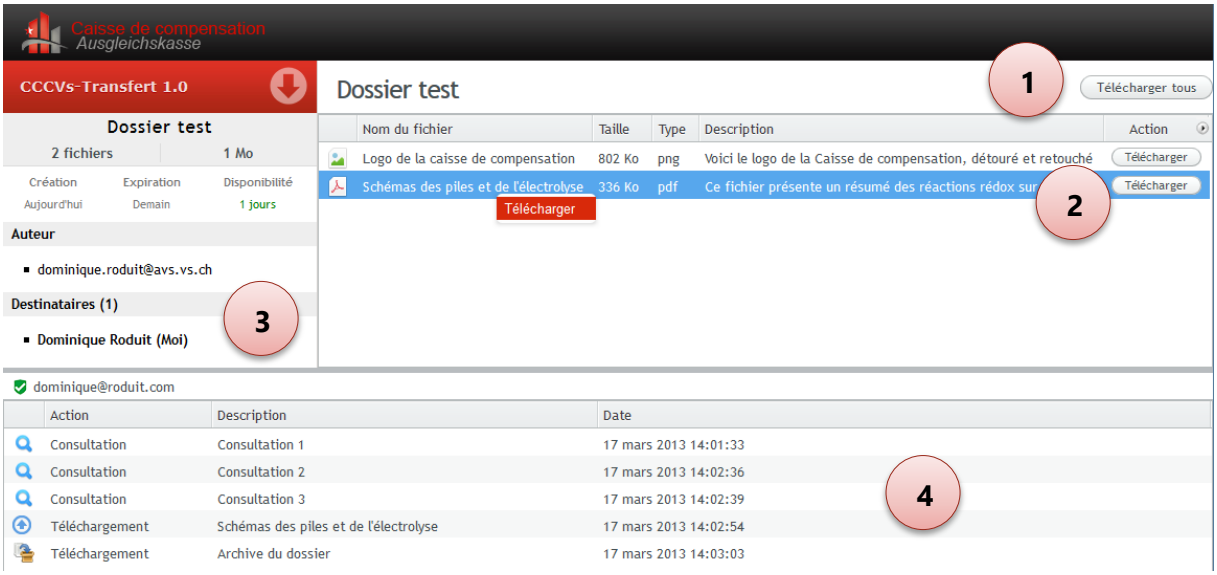
Cette dernière est également accessible depuis l'onglet « Récupérer le lien » du menu contextuel (clic droit sur un fichier ou un dossier).

Vous pouvez ensuite copier le lien (CTRL+C) pour le partager avec la personne désirée.

Téléchargement des fichiers

CCCVs-Transfert

Lorsque vous recevez par e-mail un lien vous autorisant à télécharger les fichiers d'un dossier, cliquez sur ce lien. Vous arriverez sur une page similaire à celle-ci :



1 – Vous avez la possibilité de télécharger tous les fichiers de ce dossier en une seule fois.

2 – Vous pouvez également télécharger seulement les fichiers dont vous avez besoin.

3 – Vous pouvez visualiser les informations concernant ce dossier.

4 – Vos actions sont journalisées et affichées dans cette zone. Les 3 actions distinguées sont :

- Consultation du dossier
- Téléchargement d'un fichier unique
- Téléchargement de l'archive regroupant tous les fichiers du dossier

Cette journalisation permettra à l'auteur du transfert de savoir qui a téléchargé les fichiers, qui a accédé au dossier sans télécharger de fichiers, etc…

CCCVs-Transfert

JavaDoc

Caisse de compensation du canton du Valais
Ausgleichskasse Wallis

Service informatique

Dominique Roduit

27/03/2013

dominique.roduit@avs.vs.ch

Table Of Content

Package common.component

Class Summary

AccordionMenu

Création du composant accordéon du menu qui contient les modules ContactTable et MenuFolderTable.

ContactTable

Table pour la gestion des contacts.

MenuAdmin

Menu affiché uniquement aux administrateurs de l'application.

MenuContactTable

Table pour la gestion des contacts, affichée dans le menu latéral gauche.

MenuFolderTable

Table pour la gestion des dossiers.

PanelLight

Création d'un panel dépourvu du style CSS natif de Vaadin

SQLTable

Extension du composant Table de Vaadin.

---

common.component

Class AccordionMenu

```
java.lang.Object
    |
    +--HorizontalLayout
        |
        +--common.component.AccordionMenu
```

< Constructors > < Methods >

---

public class **AccordionMenu**
extends HorizontalLayout

Création du composant accordéon du menu qui contient les modules ContactTable et MenuFolderTable.

Author:

Dominique Roduit

Constructors

AccordionMenu

```
public  AccordionMenu()
```

Methods

getMenu

```
public static Accordion getMenu()
```

Retourne l'accordéon (composant du menu)

Returns:

Menu latéral

---

getSelectedTab

```
public static java.lang.String getSelectedTab()
```

Retourne le caption de l'onglet sélectionné dans le menu

Returns:

Caption de l'onglet sélectionné du menu

---

selectTab

```
public static void selectTab(java.lang.String caption)
```

Sélectionne un onglet du menu

Parameters:

caption - Le texte de l'onglet à sélectionner

---

selectedTabChange

```
public void selectedTabChange(SelectedTabChangeEvent event)
```

common.component

Class ContactTable

```
java.lang.Object
    |
    +--Table
        |
        +--SQLTable
            |
            +--common.component.ContactTable
```

Direct Known Subclasses:
MenuContactTable

< Fields > < Constructors > < Methods >

---

public class **ContactTable**
extends SQLTable

Table pour la gestion des contacts. Cette class permet l'affichage des contacts dans un tableau. Le tableau implémente un menu, affiché sur le clic droit de la souris. Ce menu propose la suppression et l'édition des contact. En sélection multiple dans le tableau, plusieurs contacts peuvent être supprimés en même temps, mais seule la ligne sur laquelle pointe le curseur est éditée lorsqu'il s'agit d'une édition. Lorsque l'édition est choisie, les informations sont transmises vers la class ContactForm.

Author:
Dominique Roduit

Version:
1.0

Fields

ACTIONS

`protected static Action[] ACTIONS`
Liste des actions du menu contextuel affiché sur le clique droit pour ce tableau

---

ACTION_DEL

`protected static Action ACTION_DEL`
Action supprimer du menu contextuel

ACTION_EDIT

`protected static Action ACTION_EDIT`
Action éditer du menu contextuel

---

global

`protected GlobalObjects global`
Contient toutes les instances qui doivent être accessibles dans toute l'application

---

selectedItems

`protected java.lang.Object selectedItems`
Contient la/les lignes du tableau sélectionnées

Constructors

ContactTable

`public ContactTable()`

Affiche un tableau contenant la liste des contacts

Methods

actionDelete

`public void actionDelete(java.lang.Object target)`

Action exécutée sur le clique de l'option "Supprimer" du menu contextuel

Parameters:

target - (Object) Item en cours

---

actionEdit

`public void actionEdit(java.lang.Object target)`

Action exécutée sur le clique de l'option "Editer" du menu contextuel

Parameters:

target - (Object) Item en cours

---

getTableData

```
public java.lang.Object[] getTableData()
```

Sélectionn des données qui vont remplir notre tableau

Returns:

(Object[]) Tableau d'objet contenant les lignes du tableau

---

reload

```
public void reload()
```

(Re)Chargement des données du tableau

---

common.component

Class MenuAdmin

```
java.lang.Object
    |
    +--CustomComponent
         |
         +--common.component.MenuAdmin
```

< Constructors >

---

public class **MenuAdmin**
extends CustomComponent

Menu affiché uniquement aux administrateurs de l'application.

Il permet la gestion des lis

Author:

Dominique Roduit

Constructors

MenuAdmin

```
public  MenuAdmin()
```

Création des composants du menu d'administration

---

common.component

Class MenuContactTable

```
java.lang.Object
    |
    +--Table
         |
         +--SQLTable
              |
              +--ContactTable
                   |
                   +--common.component.MenuContactTable
```

< Fields > < Constructors > < Methods >

---

public class **MenuContactTable**
extends ContactTable

Table pour la gestion des contacts, affichée dans le menu latéral gauche. Cette class permet l'affichage des contacts dans un tableau. Le tableau implémente un menu, affiché sur le clic droit de la souris. Ce menu propose la suppression des contacts. En sélection multiple dans le tableau, plusieurs contacts peuvent être supprimés en même temps.

Author:

Dominique Roduit

Version:

1.0

Fields

ACTIONS

```
protected static Action[] ACTIONS
```

Liste des actions du menu contextuel affiché sur le clique droit pour ce tableau

---

ACTION_DEL

```
protected static Action ACTION_DEL
```

Action supprimer du menu contextuel

---

contactTable

```
protected ContactTable contactTable
```

Stockage de la table des contacts du corps de page

Constructors

MenuContactTable

`public` **`MenuContactTable`**`()`

Affiche un tableau contenant la liste des contacts

Methods

actionDelete

`public void` **`actionDelete`**`(java.lang.Object target)`

Action exécutée sur le clique de l'option "Supprimer" du menu contextuel

Parameters:

target - (Object) Item en cours

Overrides:

actionDelete in class ContactTable

---

getTableData

`public java.lang.Object[]` **`getTableData`**`()`

Sélectionn des données qui vont remplir notre tableau

Returns:

(Object[]) Tableau d'objet contenant les lignes du tableau

Overrides:

getTableData in class ContactTable

---

common.component

Class MenuFolderTable

```
java.lang.Object
    |
    +--Table
        |
        +--SQLTable
            |
            +--common.component.MenuFolderTable
```

< Fields > < Constructors > < Methods >

public class **MenuFolderTable**
extends SQLTable

Table pour la gestion des dossiers. Cette class permet l'affichage des dossiers dans un tableau. Le tableau implémente un menu, affiché sur le clic droit de la souris. Ce menu propose la suppression des dossiers expirés et le téléchargement des dossiers toujours disponibles.

Author:

Dominique Roduit

Version:

1.0

Fields

ACTIONS

`protected static Action[]` **`ACTIONS`**

Liste des actions du menu contextuel affiché sur le clique droit pour ce tableau

---

ACTION_DEL

`protected static Action` **`ACTION_DEL`**

Action supprimer du menu contextuel

ACTION_DOWNLOAD

`protected static Action` **`ACTION_DOWNLOAD`**

Action téléchargement

ACTION_GET_LINK

`protected static Action` **`ACTION_GET_LINK`**

Action pour la récupération du lien de téléchargement

ACTION_REMAIL

`protected static Action` **`ACTION_REMAIL`**

Action pour le renvoi d'un lien

---

global

`protected GlobalObjects` **`global`**

Contient toutes les instances qui doivent être accessibles dans toute l'application

---

selectedItems

`protected java.lang.Object` **`selectedItems`**
Contient la/les lignes du tableau sélectionnées

Constructors

MenuFolderTable

`public` **`MenuFolderTable`**`()`

Affiche un tableau contenant la liste des contacts

Methods

actionDelete

`public void` **`actionDelete`**`(java.lang.Object target)`

Action exécutée sur le clique de l'option "Supprimer" du menu contextuel

Parameters:

target - (Object) Item en cours

---

actionEdit

`public void` **`actionEdit`**`(java.lang.Object target)`

Action exécutée sur le clique de l'option "Editer" du menu contextuel

Parameters:

target - (Object) Item en cours

---

getTableData

`public java.lang.Object[]` **`getTableData`**`()`

Sélectionn des données qui vont remplir notre tableau

Returns:

(Object[]) Tableau d'objet contenant les lignes du tableau

---

loadContentForSelectedItem

`public void` **`loadContentForSelectedItem`**`()`

Charge le contenu pour l'item sélectionné

---

reload

`public void` **`reload`**`()`

(Re)Chargement des données du tableau

---

common.component

Class PanelLight

```
java.lang.Object
    |
    +--Panel
        |
        +--common.component.PanelLight
```

< Constructors > < Methods >

public class **PanelLight**
extends Panel

Création d'un panel dépourvu du style CSS natif de Vaadin

Author:

Dominique Roduit

Constructors

PanelLight

`public` **`PanelLight`**`()`

Création d'un Panel sans style CSS

Methods

setMargin

```
public void setMargin(boolean enabled)
```

Spécifications des marges

Parameters:

enabled - true = Marges activées

---

setSpacing

```
public void setSpacing(boolean enabled)
```

Spécification des marges intérieur (padding)

Parameters:

enabled - true = marges intérieur activées (padding)

---

common.component

Class SQLTable

```
java.lang.Object
    |
    +--Table
        |
        +--common.component.SQLTable
```

Direct Known Subclasses:

ContactTable, MenuFolderTable

< Fields > < Constructors > < Methods >

---

public class **SQLTable**
extends Table

Extension du composant Table de Vaadin.

Cette table contient des méthodes utiles pour le chargement de données depuis une base de données,

il faut donc l'utiliser lorsqu'on souhaite afficher les données d'une table de la base de données dans un tableau de l'interface utilisateur. Cette class est une alternative à l'add-on Vaadin "SQLContainer" qui est complexe et propose beaucoup de méthodes dont je n'ai pas l'utilité.

Contrairement au SQLContainer, les requêtes SQL exécutées pour le chargement de ce type de table contiennent directement les clauses ORDER BY, LIMIT, etc...

Author:

Dominique Roduit

Fields

PKField

```
protected java.util.ArrayList PKField
```

Constructors

SQLTable

```
public  SQLTable()
```

Création d'une table destinée à l'affichage d'un résultat d'une requête SQL

Methods

addAllItems

```
public void addAllItems(java.lang.Object[] items)
```

Ajoute tout le contenu d'un tableau d'objet (Items) dans le container

Parameters:

items - (Object[]) Résultats de la requête SQL sous forme d'items prêts à l'utilisation.

---

getValuesFromObject

```
protected java.lang.String[] getValuesFromObject(java.lang.Object value)
```

Converti un objet en tableau de String.

Utilisé pour convertir l'objet "value" en tableau de chaine de caractères

Parameters:

value - Objet à passer en paramètre

Returns:

(String[]) Tableau de string avec les valeurs

---

reload

```
public void reload(java.lang.Object[] items)
```

Mise à jour des données du tableau

Parameters:

items - Lignes de la table

Package common.component.upload

Interface Summary

UploadActionListener

Interface contenant les évènements reçus par les fichiers lors de l'envoi.

Class Summary

CustomUpload

Utilisation de la classe MultipleFileUpload, récupération et implémentation des évènements de l'interface UploadActionListener.

FileInfo

Class intermédiaire dont le but est de stocker des informations précises sur un fichier.

FileUp

Model d'un fichier pour l'intégration dans le tableau d'upload.

MultipleFileUpload

MultipleFileUpload est la class principale qui permet le téléchargement de fichiers multiple.

---

common.component.upload

Class CustomUpload

```
java.lang.Object
    |
    +--CssLayout
        |
        +--MultipleFileUpload
            |
            +--common.component.upload.CustomUpload
```

< Constructors > < Methods >

---

public class **CustomUpload**
extends MultipleFileUpload

Utilisation de la classe MultipleFileUpload, récupération et implémentation des évènements de l'interface UploadActionListener. Gestion de l'affichage de la progression et de la limitation de la taille des fichiers.

Author:

Dominique Roduit

Constructors

CustomUpload

```
public  CustomUpload(Folder folder,
                     java.util.ArrayList slctContacts)
```

Gestion des évènements qui surviennent lorsque des fichiers sont transférés

Methods

createReceiver

```
protected FileBuffer createReceiver()
```

Création du buffer qui stock les données sur le disque

Overrides:

createReceiver in class MultipleFileUpload

---

getRenamedFileList

```
public java.util.ArrayList getRenamedFileList()
```

Returns:

Obtention de la liste des fichiers tels qu'ils sont enregistrés à la fin de l'upload

---

getStatus

```
public int getStatus()
```

Returns:

Retourne le statut (0=Occupé, en envoi, 1=Libre, en attente de réception de fichiers)

---

handleFile

```
protected void handleFile(java.io.File file,
                          java.lang.String fileName,
                          java.lang.String mimeType,
                          long length)
```

Fonction appelée lorsque l'upload d'un fichier est terminé (equivalent du Listener "fileUploadFinished")

Parameters:

file - Informations sur le fichier uploadé
fileName - Nom du fichier uploadé
mimeType - Type MIME du fichier uploadé

Overrides:

handleFile in class MultipleFileUpload

---

common.component.upload

Class FileInfo

```
java.lang.Object
    |
    +--common.component.upload.FileInfo
```

< Constructors > < Methods >

public class **FileInfo**
extends java.lang.Object

Class intermédiaire dont le but est de stocker des informations précises sur un fichier.

On peut ensuite utiliser directement ce modèle pour récupérer tous les attributs du fichier dont on a besoin.

Utilisé exclusivement pour l'upload de fichier dans la class {@link MultiFileUpload}.

Author:
Dominique Roduit

Constructors

FileInfo

```
public  FileInfo(int id,
                 java.lang.String fileName,
                 long contentLength,
                 java.lang.String mimeType)
```

Création d'un model de fichier pour le stockage des informations d'un fichier dans un objet

Parameters:

id - Identifiant du fichier
fileName - Nom du fichier
contentLength - Taille du fichier
mimeType - Type MIME du fichier

Methods

getId

```
public int getId()
```

Returns:

Identifiant du fichier

---

getName

```
public java.lang.String getName()
```

Returns:

Nom du fichier

---

getSize

```
public long getSize()
```

Returns:

Taille du fichier

---

getType

```
public java.lang.String getType()
```

Returns:

Type MIME du fichier

setId

```
public void setId(int id)
```

Parameters:

id - Identifiant du fichier

---

common.component.upload

Class FileUp

```
java.lang.Object
    |
    +--common.component.upload.FileUp
```

< Constructors > < Methods >

---

public class **FileUp**
extends java.lang.Object

Model d'un fichier pour l'intégration dans le tableau d'upload. La class sert à stocker les informations sur les fichiers sélectionnés dans le but de pouvoir récupérer tous les attributs d'un fichier.

Author:

Dominique Roduit

Constructors

FileUp

```
public  FileUp(java.io.File fileDiskInfo,
               java.lang.String originalName)
```

Création d'un model de fichier pour la table d'upload

Parameters:

fileDiskInfo - Informations sur le fichier enregistré sur le disque
originalName - Nom original du fichier (tel qu'il était nommé sur le PC de l'auteur)

FileUp

```
public  FileUp(java.io.File fileDiskInfo,
               java.lang.String originalName,
               java.lang.String rename)
```

Création d'un model de fichier pour la table d'upload

Parameters:

fileDiskInfo - Informations sur le fichier enregistré sur le disque
originalName - Nom original du fichier (tel qu'il était nommé sur le PC de l'auteur)
rename - Nom du fichier renommé

Methods

getDescription

```
public java.lang.String getDescription()
```

Returns:

Description du fichier

---

getFileDiskInfo

```
public java.io.File getFileDiskInfo()
```

Obtention des informations sur le fichier enregistré sur le disque

Returns:

Informations sur le fichier enregistré

---

getId

```
public int getId()
```

Returns:

Obtention de l'identifiant du fichier dans le tableau

---

getOriginalName

```
public java.lang.String getOriginalName()
```

Obtention du nom original du fichier

Returns:

Nom du fichier original

getRename

```
public java.lang.String getRename()
```

Returns:

Nom du fichier redéfinit par l'utilisateur

setDescription

```
public void setDescription(java.lang.String description)
```

Parameters:

description - Description du fichier

setFileDiskInfo

```
public void setFileDiskInfo(java.io.File fileDiskInfo)
```

Enregistrement des informations du fichier enregistré

Parameters:

fileDiskInfo - informations du fichier enregistrél

setId

```
public void setId(int id)
```

Parameters:

id - ID du fichier dans le tableau

setOriginalName

```
public void setOriginalName(java.lang.String originalName)
```

Enregistrement du nom original du fichier

Parameters:

originalName -

setRename

```
public void setRename(java.lang.String rename)
```

Parameters:

Nom - du fichier redéfinit par l'utilisateur

common.component.upload

Class MultipleFileUpload

```
java.lang.Object
    |
    +--CssLayout
        |
        +--common.component.upload.MultipleFileUpload
```

Direct Known Subclasses:
CustomUpload

< Constructors > < Methods >

public abstract class **MultipleFileUpload**
extends CssLayout

MultipleFileUpload est la class principale qui permet le téléchargement de fichiers multiple. Elle est basée sur la class {@link MultiFileUpload} de Vaadin qui permet le téléchargement immédiat de plusieurs fichiers en parallèles. Elle affiche également les indicateurs de progression de l'envoi des fichiers ainsi qu'une zone de drag&drop pour les navigateurs qui acceptent cette technologie.

Cette class enregistre les flux directement dans des fichiers pour limiter la consommation de mémoire.

Créé des fichiers temporaires par défaut, mais cela peut être modifié avec {@link #setFileFactory(FileFactory)} (par ex. pour cibler directement le répertoire serveur) TODO Temps restant estimé et taux de transfert

Constructors

MultipleFileUpload

```
public MultipleFileUpload()
```

Methods

addUploadActionListener

`public void addUploadActionListener(UploadActionListener l)`

Ajoute l'action spécifiée {@link UploadActionListener}. Tous les {@link UploadActionListener} enregistrés seront informés des actions de téléchargement des fichiers. Il faut supprimer le Listener à la fin de l'utilisation pour prévenir les fuites de mémoire

Si le Listener est déjà enregistré, rien ne se passera.

Parameters:

l - Le Listener à ajouter

---

attach

`public void attach()`

---

createProgressIndicator

`public ProgressIndicator createProgressIndicator()`

Création d'une barre de progression

Returns:

(ProgressIndicator) Barre de progression

---

createReceiver

`protected FileBuffer createReceiver()`

---

decreaseQueueSize

`public void decreaseQueueSize(long sizeToDecrease)`

Soustrait une valeur à la taille de la liste des fichiers envoyés

Parameters:

sizeToDecrease - La valeur à soustraire

---

drop

`public void drop(DragAndDropEvent event)`

Méthode exécutée lorsqu'un fichier est déposé dans la zone prévue à cet effet

---

getAcceptCriterion

`public AcceptCriterion getAcceptCriterion()`

Retourne les critères d'acceptation d'un fichier pour l'envoi Ne pas utiliser cette méthode pour le moment, elle n'est pas encore implémentée

---

getAllowedExtentions

`public java.util.ArrayList getAllowedExtentions()`

Retourne la liste des extensions autorisées

Returns:

Extensions des fichiers autorisés

---

getAllowedFilesUploaded

`public int getAllowedFilesUploaded()`

Returns:

Nombre de fichié autorisés envoyés

---

getAreaText

`public java.lang.String getAreaText()`

Retourne le texte de la zone de drag&drop

Returns:

Texte de la zone de drag&drop

---

getFileFactory

`public FileFactory getFileFactory()`

---

getInitListFile

```
public java.util.ArrayList getInitListFile()
```

Retourne la liste des fichiers sélectionnés au départ

Returns:

Liste des fichiers sélectionnés

---

getNonAutorizedFiles

```
public java.util.ArrayList getNonAutorizedFiles()
```

Retourne la liste des fichiers non-autorisés

Returns:

Liste des fichiers non-autorisés

---

getPendingFiles

```
protected int getPendingFiles()
```

Retourne le nombre de fichiers en attente

Returns:

Nombre de fichiers en attente de téléchargement

---

getPollinInterval

```
protected int getPollinInterval()
```

Renvoi l'interval de mise à jour de la barre de progression

Returns:

Interval de mise à jour de la ProgressBar

---

getQueueSize

```
public long getQueueSize()
```

Retourne la taille des fichiers envoyés

Returns:

Taille des fichiers déjà sélectionnés

---

getUploadButtonCaption

```
public java.lang.String getUploadButtonCaption()
```

Retourne le texte du bouton qui permet de sélectionner les fichiers

Returns:

Texte du bouton Parcourir...

---

handleFile

```
protected abstract void handleFile(java.io.File file,
                                   java.lang.String fileName,
                                   java.lang.String mimeType,
                                   long length)
```

Méthode appelée obligatoirement à la fin de chaque envoi de fichier

Parameters:

file - Informations sur le fichier envoyé
fileName - Nom du fichier envoyé
mimeType - Type MIME du fichier envoyé
length - Taille du fichier envoyé

---

isDropZoneVisible

```
public boolean isDropZoneVisible()
```

Retourne la visibilité de la dropZone.

Returns:

true : dropZone visible, false : dropZone masquée.

---

isInProcess

```
public boolean isInProcess()
```

Indique si un envoi est en cours ou s'il y a encore des fichiers en attentes

Returns:

true si un fichier est encours de téléchargement ou s'il y a encore des fichiers en attente

---

removeUploadActionListener

`public void removeUploadActionListener(UploadActionListener l)`

Supprime l'action {@link UploadActionListener} spécifiée. L' {@link UploadActionListener} ne sera plus informé sur les actions de téléchargement des fichiers If the listener is not registered nothing will be do.

Parameters:

l - the listener to remove

---

setAllowedExtentions

`public void setAllowedExtentions(java.lang.String extentions)`

Fixe les extensions de fichiers qui peuvent être envoyés

Parameters:

extentions - Extensions de fichiers séparés par des virgules (exe,jpg,txt) ou * pour tous les fichiers

---

setAreaText

`public void setAreaText(java.lang.String areaText)`

Définit le texte de la zone de drag&drop

Parameters:

areaText - Texte de la zone de drag&drop. Peut contenir du code HTML

---

setDropZoneVisible

`public void setDropZoneVisible(boolean dropZoneVisible)`

Définit la visibilité de la dropZone

Parameters:

D#nit - l'attribut dropZoneVisible à la valeur spécifiée

---

setEnableUploadButton

`public void setEnableUploadButton(boolean enable)`

Active/Désactive le bouton pour choisir les fichiers

Parameters:

enable(true) - Activé (false) Désactivé

---

setEnabledUploadDropZone

`public void setEnabledUploadDropZone(boolean enable)`

Active/Désactive la zone de Drag/Drop

Parameters:

enable - (true) Activé (false) Désactivé

---

setFileFactory

`public void setFileFactory(FileFactory fileFactory)`

---

setMaxQueueSize

`public void setMaxQueueSize(int SizeLimit)`

Fixe la taille limite pour le total des fichiers

Parameters:

SizeLimit - Taille maximum en octet

---

setRootDirectory

`public void setRootDirectory(java.lang.String directoryWhereToUpload)`

Une méthode d'assistance pour définir DirectoryFileFactory avec le chemin du répertoire spécifié

Parameters:

directoryWhereToUpload - Répertoire dans lequel envoyer les fichiers

---

setUploadButtonCaption

`public void setUploadButtonCaption(java.lang.String uploadButtonCaption)`

Fixe le texte du bouton Parcourir...

Parameters:

uploadButtonCaption - Texte du bouton

---

supportsFileDrops

```
protected boolean supportsFileDrops()
```

Indique si le drag&drop est supporté par le navigateur ou non.

Les navigateurs qui prennent en charge cette fonctionnalité sont :

- Chrome
- Firefox
- Safari

Returns:

true : le drag&drop est activé.

---

common.component.upload

Interface UploadActionListener

< Methods >

---

public interface **UploadActionListener**

Interface contenant les évènements reçus par les fichiers lors de l'envoi.

Son implémentation permet de détecter le début et la fin d'un transfert, de détecter les erreurs et de suivre la progression des fichiers.

Author:

steffen.c-on, Dominique Roduit

Version:

18.03.2013

Methods

fileUploadComplete

```
public void fileUploadComplete(java.util.ArrayList fileList)
```

Appelé lorsque tout les fichiers sont envoyés avec succès

Parameters:

fileList - Liste des fichiers envoyés

---

fileUploadError

```
public void fileUploadError(java.lang.String filename,
                            int pendingFiles)
```

Called if a upload of a file is finished with an error.

Parameters:

filename - name of the file which is aborted
pendingFiles - number of pending files

---

fileUploadFinished

```
public void fileUploadFinished(int idFile,
                               java.lang.String filename,
                               java.io.File file,
                               int pendingFiles)
```

Called if a upload of a file is finished successful.

Parameters:

filename - name of the file which is currently finished
pendingFiles - number of pending files

---

fileUploadProgress

```
public void fileUploadProgress(int idFile,
                               java.lang.String filename,
                               int pendingFiles,
                               float progress)
```

Appelé durant la progression de l'upload d'un fichier

Parameters:

idFile - Identifiant du fichier
filename - Nom du fichier
pendingFiles - Nombre de fichiers en attentes
progress - Progression (de 0 à 1)

---

fileUploadStarted

```
public void fileUploadStarted(int idFile,
                              java.lang.String filename,
                              int pendingFiles)
```

Called if a upload of a file is started.

Parameters:

filename - name of the file which is currently uploading
pendingFiles - number of pending files (without the currently uploading file)

onSelectedFiles

```
public void onSelectedFiles(int error)
```

Appelé lorsque les fichiers sont sélectionnés (ou déposés dans la fenêtre)

Parameters:

error - Code d'erreur retourné (s'il y en a une)

Package form

Class Summary

ContactForm

Cette class permet l'ajout et la mise à jour des contacts via un formulaire affiché dans une sous-fenêtre.

WizFoldDest

2e vue de l'assistant pour la création d'un dossier.

WizFoldNew

Premier Wizard de l'assistant de création d'un dossier.

WizFoldUpload

3e vue de l'assistant de création d'un dossier.

---

form

Class ContactForm

```
java.lang.Object
    |
    +--CustomComponent
        |
        +--form.ContactForm
```

< Constructors > < Methods >

public class **ContactForm**
extends CustomComponent

Cette class permet l'ajout et la mise à jour des contacts via un formulaire affiché dans une sous-fenêtre.
Un constructeur permet l'ajout, un autre permet la mise à jour.

Author:

Dominique

Constructors

ContactForm

```
public ContactForm()
```

Affiche un formulaire pour l'ajout de contact

ContactForm

```
public  ContactForm(java.lang.String PK)
```

Affiche un formulaire pour l'édition d'un contact

Parameters:

PK - La clé primaire du contact à éditer

Methods

getButtonApply

```
public Button getButtonApply()
```

Retourne une instance du bouton "Ajouter" de la sous-fenêtre

Returns:

Instance du bouton "Ajouter"

---

form

Class WizFoldDest

```
java.lang.Object
    |
    +--CustomComponent
        |
        +--form.WizFoldDest
```

< Constructors >

---

public class **WizFoldDest**
extends CustomComponent

2e vue de l'assistant pour la création d'un dossier.

Cette vue affiche un tableau ainsi qu'une liste déroulante contenant la liste de tous les contacts de l'utilisateur. L'utilisateur peut choisir lesquels contacts il souhaite ajouter comme destinataires. Cette class transmet les informations sélectionnées à la vue suivante ({@link WizFoldUpload}).

Author:

Dominique Roduit

Constructors

WizFoldDest

```
public  WizFoldDest(Folder folder)
```

Affiche un formulaire pour l'ajout de destinataires à un dossier

Parameters:

folder - Informations sur le dossier, définies dans le wizard 1

---

form

Class WizFoldNew

```
java.lang.Object
    |
    +--CustomComponent
        |
        +--form.WizFoldNew
```

< Constructors >

---

public class **WizFoldNew**
extends CustomComponent

Premier Wizard de l'assistant de création d'un dossier.

Il permet la définition du nom du dossier, de sa description, et le choix de sa durée de vie.

Aucune information n'est enregistrée dans ce Wizard. Toutes les données saisies par l'utilisateur sont transmises aux vues suivantes ({@link WizFoldDest}, {@link WizFoldUpload}).

Author:

Dominique Roduit

Constructors

WizFoldNew

```
public  WizFoldNew()
```

Affiche un formulaire pour l'ajout de contact

---

form

Class WizFoldUpload

```
java.lang.Object
    |
    +--CustomComponent
         |
         +--form.WizFoldUpload
```

< Constructors > < Methods >

public class **WizFoldUpload**
extends CustomComponent

3e vue de l'assistant de création d'un dossier.

Ce Wizard permet l'upload des fichiers, le stockage dans un tableau et le renommage des fichiers envoyés.

Sur le clic du bouton "Terminer", toutes les informations sont enregistrées dans la base de données, les mails envoyés aux destinataires et l'archive contenant l'ensemble des fichiers du dossier est générée.

Author:
Dominique Roduit

Constructors

WizFoldUpload

```
public  WizFoldUpload(Folder folder,
                      java.util.ArrayList slctContacts)
```

Enregistrement des valeurs récoltées par les Wizard précédent et création du formulaire

Parameters:

folder - Informations sur le dossier, définies dans le wizard 1
slctContacts - Destinataires sélectionnés dans le wizard 2

Methods

getBtCancel

```
public static Button getBtCancel()
```

Retourne le bouton "Annuler"

Returns:

Bouton "Annuler"

---

getBtFinish

```
public static Button getBtFinish()
```

Retourne le bouton "Terminer"

Returns:

Bouton "Terminer"

---

getBtRename

```
public static Button getBtRename()
```

Returns:

Bouton "Renommer"

---

getLblSize

```
public static Label getLblSize()
```

Returns:

Label qui contient la taille envoyée/restante

---

getSlctFileLbl

```
public static Label getSlctFileLbl()
```

Retourne le label indiquant de sélectionner les fichier

Returns:

Label indiquant de sélectionner les fichiers

---

getSuperIndicator

```
public static ProgressIndicator getSuperIndicator()
```

Returns:

Indicateur de progression de l'envoi des fichiers

Package global

Class Summary

Global

Class de stockage des champs Globaux (qui doivent être accessibles pour toutes les class de l'application).

GlobalObjects

Class stockant tous les objets utilisés souvent, qui doivent être accessible n'importe où dans l'application.

Module

Création d'un module dans le but de l'afficher sur la page.

UserSession

Stockage des informations d'un utilisateur lors de sa connexion.

---

global

Class Global

```
java.lang.Object
    |
    +--global.Global
```

< Fields > < Constructors > < Methods >

---

public class **Global**
extends java.lang.Object

Class de stockage des champs Globaux (qui doivent être accessibles pour toutes les class de l'application).

Elle contient les méthodes de récupération des paramètres et des traductions de l'application. Toutes les méthodes et champs de la class sont statiques.

Author:
Dominique Roduit

Fields

APP_LANGUAGE

`public static java.lang.String` **`APP_LANGUAGE`**
Langue par défaut de l'interface

IMG_APPROVED

`public static final java.lang.String` **`IMG_APPROVED`**
Image affichée dans le footer à côté de l'adresse de l'utilisateur connecté

IMG_ARCHIVE_DOWNLOAD

`public static final java.lang.String` **`IMG_ARCHIVE_DOWNLOAD`**
Image illustrant une archive téléchargée

IMG_CONTACT

`public static final java.lang.String` **`IMG_CONTACT`**
Image contact

IMG_DOWNLOAD

`public static final java.lang.String` **`IMG_DOWNLOAD`**
Image illustrant un téléchargement

IMG_FLAG_COUNTRY

`public static java.lang.String` **`IMG_FLAG_COUNTRY`**

IMG_FOLDER_ADD

`public static final java.lang.String` **`IMG_FOLDER_ADD`**
Image d'ajout d'un dossier

IMG_MAIN_LOGO

`public static final java.lang.String` **`IMG_MAIN_LOGO`**
Logo affiché dans le header du layout principal

IMG_SEARCH

`public static final java.lang.String` **`IMG_SEARCH`**
Image illustrant une consultation

PATH_THEME_RESSOURCES

`public static final java.lang.String` **`PATH_THEME_RESSOURCES`**
Répertoire contenant les ressources (images) du thème

PATH_THEME_RESSOURCES_HTML

`public static final java.lang.String` **`PATH_THEME_RESSOURCES_HTML`**
Répertoire contenant les ressources du thème.

A utiliser lorsqu'on écris l'image en HTML

ex. :

UPLOAD_DIR

`public static java.lang.String` **`UPLOAD_DIR`**
Paramètre très important qui donne le répertoire dans lequel enregistrer les fichier

Initialisation dans la class principal {@link CccvsTransfert}

URL

`public static java.net.URL` **`URL`**
URL de l'application. Seulement la partie http://domaine.ch:port/monApplication. Sans les paramètres de l'URL.

URLParameters

`public static java.lang.String` **`URLParameters`**
Contient les paramètres de l'URL s'il y en a

---

browser

`public static WebBrowser` **`browser`**
Contient les paramètres du navigateur tel que le nom, la version et la langue

---

hashParm

`public static final java.util.HashMap` **`hashParm`**
HashMap qui contient tous les paramètres

---

hashTranslate

`public static final java.util.HashMap` **`hashTranslate`**
HashMap qui contient toutes les traductions

---

isDev

`public static boolean` **`isDev`**
Indique si l'application est exécutée sur un poste de développement ou de production

Constructors

Global

`public` **`Global`**`()`

Methods

getParm

`public static java.lang.String` **`getParm`**`(java.lang.String key)`

Retourne un paramètre de l'application enregistré dans la base de données.

Parameters:

key - La clé pour trouver le paramètre

Returns:

(String) Paramètre de la base de données

---

i

`public static java.lang.String` **`i`**`(java.lang.String key)`

Fonction "Intertionalize". Elle permet la traduction des textes de l'application

Parameters:

key - Mot clé du texte à afficher

Returns:

Le mot traduit dans la langue en cours

---

reloadTranslations

`public static void` **`reloadTranslations`**`()`

Rechargement du tableau contenant les traductions

global

Class GlobalObjects

```
java.lang.Object
    |
    +--global.GlobalObjects
```

< Constructors > < Methods >

public class **GlobalObjects**
extends java.lang.Object

Class stockant tous les objets utilisés souvent, qui doivent être accessible n'importe où dans l'application.

Attention cependant de ne pas utiliser un objet avant qu'il n'ait été créé et enregistré dans le champ qui lui est approprié. À la différence de la class Global, les méthodes ne sont pas statiques, la classe doit être instanciée pour que ses méthodes soient utilisables.

Author:
Dominique Roduit

Constructors

GlobalObjects

`public` **`GlobalObjects`**`()`

Methods

getBtAddContact

`public Button` **`getBtAddContact`**`()`

Obtention du bouton "Ajouter un contact" {@link ContactModule}

Returns:

Bouton "Ajouter un contact"

getBtAddFolder

`public Button` **`getBtAddFolder`**`()`

Obtention du bouton "Nouveau dossier" dossier

Returns:

Bouton "Nouveau dossier"

getMainLayout

`public` MainLayout **`getMainLayout`**`()`

Obtention du layout principal de l'application. {@link MainLayout}

Returns:

Layout principal de l'application

getMenuAdmin

`public` MenuAdmin **`getMenuAdmin`**`()`

Returns:

Le menu d'administration

getMenuContactTable

`public` MenuContactTable **`getMenuContactTable`**`()`

Obtention du tableau des contacts intégré au menu {@link MenuContactTable}

Returns:

Tableau des contacts dumenu

getMenuFolderTable

`public` MenuFolderTable **`getMenuFolderTable`**`()`

Obtention du tableau des dossiers intégré au menu

Returns:

Tableau des dossiers du menu

getTableContact

`public` ContactTable **`getTableContact`**`()`

Obtention du tableau des contacts. {@link ContactTable}

Returns:

Tableau des contacts

getUploadModule

`public CustomUpload getUploadModule()`

Returns:

the uploadModule

---

getUploadModuleLayout

`public VerticalLayout getUploadModuleLayout()`

Returns:

the uploadModuleLayout

---

getUploadTable

`public Table getUploadTable()`

Obtention du tableau des fichiers uploadés.

Returns:

Tableau des fichiers uploadés.

---

getWinGetLink

`public Window getWinGetLink()`

Returns:

Fenêtre pour la récupération du lien de téléchargement

---

getWinReMailLink

`public Window getWinReMailLink()`

Returns:

Fenêtre de reditribution du lien d'accès au dossier

---

getWindowContact

`public Window getWindowContact()`

Obtention de la fenêtre contenant le formulaire d'ajout/édition de contact. {@link ContactModule}

Returns:

Fenêtre d'ajout/édition de contact

---

getWindowFolder

`public Window getWindowFolder()`

Obtention de la fenêtre de création d'un dossier

Returns:

Fenêtre de création d'un dossier

---

openWindowGetLink

`public void openWindowGetLink(java.lang.String filename)`

Ouvre la fenêtre de récupération du lien de téléchargement

Parameters:

filename - Nom du fichier pour lequel on veux obtenir le lien

---

setBtAddContact

`public void setBtAddContact(Button btAddContact)`

Stockage en global du bouton "Ajouter un contact". {@link ContactModule}

Parameters:

btAddContact - Bouton "Ajouter un contact"

---

setBtAddFolder

`public void setBtAddFolder(Button btAddFolder)`

Stockage en global du bouton "Nouveau dossier".

Parameters:

btAddFolder - Bouton "Nouveau dossier"

---

setMainLayout

`public void` **`setMainLayout`**`(MainLayout mainLayout)`

Stockage en global du layout principal de l'application. Définit dans {@link CccvsTransfert}

Parameters:

mainLayout - Layout principal de l'application

---

setMenuAdmin

`public void` **`setMenuAdmin`**`(MenuAdmin menuAdmin)`

Parameters:

menuAdmin - Menu d'administration

---

setMenuContactTable

`public void` **`setMenuContactTable`**`(MenuContactTable menuContactTable)`

Stockage en global du tableau des contacts intégré au menu. {@link MenuContactTable}

Parameters:

menuContactTable - Tableau des contacts du menu

---

setMenuFolderTable

`public void` **`setMenuFolderTable`**`(MenuFolderTable menuFolderTable)`

Stockage en global du tableau des dossiers intégré au menu

Parameters:

menuFolderTable - Tableau des dossiers du menu

---

setTableContact

`public void` **`setTableContact`**`(ContactTable tableContact)`

Stockage en global du tableau des contacts. {@link ContactTable}

Parameters:

tableContact - Tableau des contacts

---

setUploadModule

`public void` **`setUploadModule`**`(CustomUpload uploadModule)`

Parameters:

uploadModule - the uploadModule to set

---

setUploadModuleLayout

`public void` **`setUploadModuleLayout`**`(VerticalLayout uploadModuleLayout)`

Parameters:

uploadModuleLayout - the uploadModuleLayout to set

---

setUploadTable

`public void` **`setUploadTable`**`(Table uploadTable)`

Stockage de la table d'upload en global

Parameters:

uploadTable - Table d'upload

---

setWinGetLink

`public void` **`setWinGetLink`**`(Window winGetLink)`

Parameters:

winGetLink - Fenêtre pour la récupération du lien de téléchargement

---

setWinReMailLink

`public void` **`setWinReMailLink`**`(Window winReMailLink)`

Parameters:

winReMailLink - Fenêtre de reditribution du lien d'accès au dossier

---

setWindowContact

```
public void setWindowContact(Window windowContact)
```

Stockage en global de la fenêtre d'ajout/édition de contact

Parameters:

windowContact - Fenêtre d'ajout/édition de contact

---

setWindowFolder

```
public void setWindowFolder(Window windowFolder)
```

Stockage de la fenêtre de création d'un dossier

Parameters:

windowFolder - Fenêtre de création d'un dossier

---

global

Class Module

```
java.lang.Object
    |
    +--CustomComponent
        |
        +--global.Module
```

< Constructors > < Methods >

---

public class **Module**
extends CustomComponent

Création d'un module dans le but de l'afficher sur la page.

Un module se compose de deux parties :

- 1. Contenu du ruban = Zone haute du corps de page qui contient les boutons et les titres
- 2. Contenu du corps de page = Contenu principal affiché dans la partie principale de l'application
- Exemple d'appel : {@code Module MODULE_TRANSFERT = new Module(TransfertModule.getBodyRibbonContent(), TransfertModule.getBodyContent()); }

Author:

Dominique Roduit

Constructors

Module

```
public  Module(Component ribbonContent,
               Component bodyContent)
```

Création d'une instance d'un module

Parameters:

ribbonContent - Contenu du ruban
bodyContent - Contenu du corps de page

Methods

getBodyContent

```
public Component getBodyContent()
```

Obtention du contenu du corps de page

Returns:

Contenu du corps de page

---

getRibbonContent

```
public Component getRibbonContent()
```

Obtention du contenu du ruban

Returns:

Contenu du ruban

---

setBodyContent

```
public void setBodyContent(Component bodyContent)
```

Enregistre le contenu du corps de page

Parameters:

bodyContent - Contenu du corps de page

---

setRibbonContent

```
public void setRibbonContent(Component ribbonContent)
```

Enregistrement du contenu du ruban

Parameters:

ribbonContent - Contenu du ruban

---

global

Class UserSession

```
java.lang.Object
    |
    +--global.UserSession
```

< Constructors > < Methods >

---

public class **UserSession**
extends java.lang.Object

Stockage des informations d'un utilisateur lors de sa connexion.

Cette class est comparable à une variable de session en PHP.

Lors du rechargement de la page, les données sont conservées grâce à un ThreadLocal.

Si l'application est redémarrée, la session est détruite.

Author:

Dominique Roduit

Constructors

UserSession

```
public  UserSession()
```

Methods

getID

```
public static int getID()
```

Retourne la clé primaire de l'utilisateur

Returns:

Clé primaire de l'utilisateur

---

getLangue

```
public static java.lang.String getLangue()
```

Retourne la langue définit par l'utilisateur ou par son navigateur si celui-ci ne l'a jamais modifiée

Returns:

Langue de l'utilisateur

---

getMail

```
public static java.lang.String getMail()
```

Retourne l'adresse e-mail de l'utilisateur connecté

Returns:

Adresse e-mail

---

isAdmin

```
public static boolean isAdmin()
```

Returns:

Indique si l'utilisateur est un administrateur

---

isInternal

```
public static boolean isInternal()
```

Indique si l'utilisateur est un interne ou non

Returns:

true si l'utilisateur est un interne

---

setAdmin

```
public static void setAdmin(boolean admin)
```

Parameters:

D#nit - l'utilisateur comme administrateur ou non

---

setID

`public static void **setID**(int iD)`

Enregistre la clé primaire de l'utilisateur

Parameters:

iD - Clé primaire de l'utilisateur

---

setInternal

`public static void **setInternal**(boolean internal)`

Enregistre si l'utilisateur est un interne ou non

Parameters:

internal - true si l'utilisateur est un interne

---

setLangue

`public static void **setLangue**(java.lang.String langue)`

Enregistre la langue choisie par l'utilisateur

Parameters:

langue - Langue choisie par l'utilisateur

---

setMail

`public static void **setMail**(java.lang.String mail)`

Enregistre l'adresse e-mail de l'utilisateur connecté

Parameters:

mail - Adresse e-mail

Package main

Class Summary

CccvsTransfert

Application CCCVs-Transfert

Class principale de l'application qui instancie les composants racines (Fenêtres et Layouts de base).

ConnexionLayout

IMPORTANT !

MainLayout

IMPORTANT !

---

main

Class CccvsTransfert

```
java.lang.Object
    |
    +--Application
        |
        +--main.CccvsTransfert
```

< Fields > < Constructors > < Methods >

---

public class **CccvsTransfert**
extends Application

Application CCCVs-Transfert

Class principale de l'application qui instancie les composants racines (Fenêtres et Layouts de base).

C'est cette classe qui est appelée au lancement de l'application.

Elle implémente un ThreadLocal pour conserver l'état de l'application lors d'un rechargement de la page.

Si l'application est redémarrée (?restartApplication), le ThreadLocal est réinitialisé.

Author:

Roduit Dominique

Version:

1.0

2013-02-28

Fields

mainWindow

`public static Window` **`mainWindow`**

Fenêtre principale qui doit être accessible depuis n'importe ou. Pas de getter/setter parce qu'on ne peut pas le faire pour la fenêtre racine

Constructors

CccvsTransfert

`public` **`CccvsTransfert`**`()`

Methods

getGlobalMethod

`public static` `GlobalObjects` **`getGlobalMethod`**`()`

Méthode très importante qui renvoie les objets stockés en global

Returns:

(GlobalMethod) Instance stockées dans la class GlobalMethod

---

getInstance

`public static` `CccvsTransfert` **`getInstance`**`()`

Commence l'implémentation du ThreadLocal

Returns:

Application enregistrée dans le threadLocal

---

getSystemMessages

`public static SystemMessages` **`getSystemMessages`**`()`

Configuration des différents messages du système VAADIN, par exemple, lorsque les cookies sont désactivés ou que la session a expirée.

Returns:

Messages personnalisés

---

init

`public void` **`init`**`()`

Méthode appelée à l'initialisation de l'application.

Cette méthode est rappelée chaque fois que le paramètre URL ?restartApplication existe.

---

loadModule

`public static void` **`loadModule`**`(``Module`` module)`

Charge un module dans l'application

Parameters:

module - (Module) Le module à charger

---

onRequestEnd

```
public void onRequestEnd(HttpServletRequest request,
                         HttpServletResponse response)
```

Méthode appelée à la fin de chaque requête

---

onRequestStart

```
public void onRequestStart(HttpServletRequest request,
                           HttpServletResponse response)
```

Méthode appelée avant le chargement de l'application, elle permet de rétablir l'état de l'application avant le rechargement de la page.

---

resetContent

`public static void` **`resetContent`**`()`

Supprime le contenu de la page en cours

---

setInstance

`public static void` **`setInstance`**`(``CccvsTransfert`` application)`

Remplace l'état en cours de l'application par celui enregistrée dans le Thread

Parameters:

application - L'application à remplacer

main

Class ConnexionLayout

```
java.lang.Object
    |
    +--CustomComponent
        |
        +--main.ConnexionLayout
```

< Constructors > < Methods >

public class **ConnexionLayout**
extends CustomComponent

IMPORTANT !

Construction du composant de connexion à l'application.

Ce composant personnalisé est un layout qui contient le composant de connexion à l'interface utilisateur.
La class est instanciée depuis {@link CccvsTransfert}.

Author:
Dominique Roduit

Version:
1.0.0

2013-03-01

Constructors

ConnexionLayout

`public` **`ConnexionLayout`**`()`

Construction des 3 niveaux de bases (top, middle, bottom)

Methods

getButtonConnexion

`public Button` **`getButtonConnexion`**`()`

Retourne le bouton de connexion

Returns:

Bouton de connexion du formulaire

getConnexionState

`public java.lang.Boolean` **`getConnexionState`**`()`

Retourne l'état de la connexion

Returns:

true si la connexion est ok

getMainLayout

`public HorizontalLayout` **`getMainLayout`**`()`

Retourne le layout racine du composant

Returns:

Layout racine (CompositionRoot)

getTextFieldID

`public TextField` **`getTextFieldID`**`()`

Retourne le textField d'identification

Returns:

TextField d'identification

getWindowConnexion

`public Window` **`getWindowConnexion`**`()`

Retourne la fenêtre flottante contenant le contenu

Returns:

Fenêtre flottante

setConnexionState

`public void` **`setConnexionState`**`(java.lang.Boolean connexionState)`

Set l'état de la connexion

Parameters:

connexionState - Etat de la connexion

main

Class MainLayout

```
java.lang.Object
    |
    +--CustomComponent
         |
         +--main.MainLayout
```

< Constructors > < Methods >

public class **MainLayout**
extends CustomComponent

IMPORTANT !

Construction du composant principal de l'application.

Ce composant personnalisé est un layout qui contient l'interface utilisateur. Il est complexe et composé de plusieurs parties distinctes (Haut, Milieu, Bas). Chaque partie est dotée d'un menu. Cette class ne contient que les blocs et dispositions, le contenu est chargé depuis la class principale {@link CccvsTransfert}.

Author:
Dominique Roduit

Version:
1.0.0

2013-03-01

Constructors

MainLayout

`public` **`MainLayout`**`()`

Assemblage des layouts par l'appel des méthodes de création des composants

Methods

buildBottomLayout

`public void` **`buildBottomLayout`**`()`

Construction du layout du bas (footer)

---

getBody

`public VerticalLayout` **`getBody`**`()`

Retourne le corps de page dans lequel on va insérer notre contenu

Returns:
Corps de page

---

getBodyRibbon

`public HorizontalLayout` **`getBodyRibbon`**`()`

Retourne le ruban du corps de page

Returns:
Ruban du corps de page

---

getBodyRibbonWrapper

`public HorizontalLayout` **`getBodyRibbonWrapper`**`()`

Returns:
Zone racine du ruban du corps de page

---

getBtChangeLanguage

`public Button` **`getBtChangeLanguage`**`()`

Retourne le bouton de changement de langue

Returns:
Bouton de changement de langue

---

getFooter

`public VerticalLayout` **`getFooter`**`()`

Retourne la zone de téléchargement dans le footer

Returns:
Zone de téléchargement du footer

---

getHeaderLogo

`public HorizontalLayout` **`getHeaderLogo`**`()`

Retourne la zone du header contenant le logo

Returns:

Zone du header contenant le logo

---

getHeaderMenu

`public HorizontalLayout` **`getHeaderMenu`**`()`

Retourne le menu du header

Returns:

Menu du header

---

getMenu

`public VerticalLayout` **`getMenu`**`()`

Retourne la zone d'affichage dynamique du menu

Returns:

Menu latéral

---

getMenuRibbon

`public HorizontalLayout` **`getMenuRibbon`**`()`

Retourne le ruban du menu

Returns:

Ruban du menu

---

restoreRibbonHeight

`public void` **`restoreRibbonHeight`**`()`

Restaure la taille initiale du ruban

---

setFooterEnabled

`public void` **`setFooterEnabled`**`(boolean enabled)`

Affiche/Masque le footer

Parameters:

enabled - true : affiché, false : désactivé

---

setMenuWidth

`public void` **`setMenuWidth`**`(java.lang.String width)`

Fixe la taille du menu latéral

Parameters:

width - Taille du menu en px

Package model

Class Summary

ActionJournalFold

Modèle pour le stockage d'actions journalisées sur les dossiers

Table : trans_journal_folders

Contact

Modèle de contact.

Files

Modèle d'un fichier (Table trans_files).

Folder

Modèle d'un dossier de transfert (Table trans_folders).

JournalCon

Modèle d'une action de connexion journalisée (Table trans_journal_connections).

User

Modèle d'un utilisateur (Table trans_users).

---

model

Class ActionJournalFold

```
java.lang.Object
    |
    +--model.ActionJournalFold
```

< Constructors > < Methods >

---

public class **ActionJournalFold**
extends java.lang.Object

Modèle pour le stockage d'actions journalisées sur les dossiers

Table : trans_journal_folders

Author:

Dominique Roduit

Constructors

ActionJournalFold

```
public  ActionJournalFold(int PKNoJourFolder,
                          java.lang.String action_type,
                          Files action_file,
                          Folder action_folder,
                          java.lang.String action_date,
                          Contact action_contact)
```

Création d'un objet pour une action journalisée exécutée sur un dossier

Parameters:

action_type - Type d'action
action_description - Description de l'action
action_file - Fichier concerné par l'action
action_folder - Dossier concerné par l'action
action_date - Date de l'action
action_contact - Contact qui déclenche l'action

ActionJournalFold

```
public  ActionJournalFold(java.lang.String Type,
                          java.lang.String Description)
```

Création d'un objet pour une action journalisée exécutée sur un dossier

Parameters:

Type - Type d'action
Description - Description de l'action

Methods

getAction_contact

```
public Contact getAction_contact()
```

Returns:

Contact qui déclenche l'action

---

getAction_date

```
public java.lang.String getAction_date()
```

Returns:

Date de l'action

---

getAction_description

`public java.lang.String` **`getAction_description`**`()`

Obtention de la description de l'action effectuée

Returns:

Description de l'action effectuée

---

getAction_file

`public Files` **`getAction_file`**`()`

Returns:

Fichier concerné par l'action

---

getAction_folder

`public Folder` **`getAction_folder`**`()`

Returns:

Dossier concerné par l'action

---

getAction_type

`public java.lang.String` **`getAction_type`**`()`

Obtention du type d'action

Returns:

Type d'action

---

getPKNoJourFolder

`public int` **`getPKNoJourFolder`**`()`

Returns:

Clé primaire de l'action

---

setAction_contact

`public void` **`setAction_contact`**`(Contact action_contact)`

Parameters:

action_contact - Contact qui déclenche l'action

---

setAction_date

`public void` **`setAction_date`**`(java.lang.String action_date)`

Parameters:

action_date - Date de l'action

---

setAction_description

`public void` **`setAction_description`**`(java.lang.String action_description)`

Enregistrement de la description de l'action effectuée

Parameters:

action_description - Description de l'action effectuée

---

setAction_file

`public void` **`setAction_file`**`(Files action_file)`

Parameters:

action_file - Fichier concerné par l'action

---

setAction_folder

`public void` **`setAction_folder`**`(Folder action_folder)`

Parameters:

action_folder - Dossier concerné par l'action

---

setAction_type

```
public void setAction_type(java.lang.String action_type)
```

Enregistrement du type d'action

Parameters:

action_type - Type d'action

---

setPKNoJourFolder

```
public void setPKNoJourFolder(int pKNoJourFolder)
```

Parameters:

pKNoJourFolder - Clé primaire de l'action

---

model

Class Contact

```
java.lang.Object
    |
    +--model.Contact
```

All Implemented Interfaces:

java.io.Serializable

< Constructors > < Methods >

---

public class **Contact**
extends java.lang.Object
implements java.io.Serializable

Modèle de contact.

Cette class correspond aux champs de la base de données pour la table tbl_contacts.

Author:

Dominique Roduit

Constructors

Contact

```
public Contact()
```

Contact

```
public Contact(int PKNoContact,
               int FKNoUser,
               java.lang.String name,
               java.lang.String mail,
               java.lang.String creation_date,
               boolean internal)
```

Contructeur du model pour le stockage d'informations sur un contact

Parameters:

PK - Clé primaire du contact
FKNoUser - Clé étrangère vers l'utilisateur qui détient ce contact dans sa liste
name - Nom du contact
mail - Adresse e-mail du contact
creation_date - Date de créatin du contact
internal - true si le contact est un interne

Contact

```
public Contact(int PK,
               java.lang.String name,
               java.lang.String mail,
               boolean internal)
```

Constructeur du model pour le stockage d'informations sur un contact

Parameters:

PK - Clé primaire du contact
name - Nom du contact
mail - Adresse e-mail du contact
internal - true si le contact est un interne

Methods

getCreation_date

```
public java.lang.String getCreation_date()
```

Returns:

Date de création du contact

---

getFKNoUser

`public int` **`getFKNoUser`**`()`

Obtention de la clé étrangère de l'utilisateur détenant le contact

Returns:

Clé étrangère de l'utilisateur

---

getMail

`public java.lang.String` **`getMail`**`()`

Obtention de l'adresse e-mail du contact

Returns:

Adresse e-mail du contact

---

getName

`public java.lang.String` **`getName`**`()`

Obtention du nom du contact

Returns:

Nom du contact

---

getPK

`public int` **`getPK`**`()`

Obtention de la clé primaire du contact

Returns:

Cél primaire du contact

---

isInternal

`public boolean` **`isInternal`**`()`

Indique si le contact est un interne ou non

Returns:

true si le contact est un interne

---

reset

`public void` **`reset`**`()`

Vidage des champs pour l'annulation des valeurs entrées dans le formulaire d'ajout/édition

---

setCreation_date

`public void` **`setCreation_date`**`(java.lang.String creation_date)`

Parameters:

creation_date - Date de création du contact

---

setFKNoUser

`public void` **`setFKNoUser`**`(int fKNoUser)`

Définition de la clé étrangère de l'utilisateur détenant le contact

Parameters:

fKNoUser - Clé étrangère de l'utilisateur

---

setInternal

`public void` **`setInternal`**`(boolean internal)`

Définit si le contact est un interne ou non

Parameters:

internal - true si le contact est un interne

---

setMail

`public void` **`setMail`**`(java.lang.String email)`

Définition de l'adresse e-mail du contact

Parameters:

email - Adresse e-mail du contact

---

setName

```
public void setName(java.lang.String name)
```

Définition du nom du contact

Parameters:

name - Nom du contact

---

setPK

```
public void setPK(int pK)
```

Définition de la clé primaire du contact

Parameters:

pK - Clé primaire du contact

---

model

Class Files

```
java.lang.Object
    |
    +--model.Files
```

< Constructors > < Methods >

---

public class **Files**
extends java.lang.Object

Modèle d'un fichier (Table trans_files).

Permet le stockage des informations sur un fichier dans un objet.

Author:

Dominique Roduit

Constructors

Files

```
public  Files(int PKNoFile,
              int FKNoFolder,
              java.lang.String file_name,
              java.lang.String file_rename,
              long file_size,
              java.lang.String file_extension,
              java.lang.String file_description)
```

Constructeur du model pour le stockage d'un model de fichier

Parameters:

PKNoFile - Clé primaire du fichier
FKNoFolder - Clé étrangère du dossier
file_name - Nom du fichier
file_rename - Nom du fichier spécifié par l'utilisateur
file_size - Taille du fichier
file_extension - Extension du fichier
file_description - Description du fichier

Files

```
public  Files(int PKNoFile,
              int FKNoFolder,
              java.lang.String file_name,
              java.lang.String file_rename,
              long file_size,
              java.lang.String file_extension,
              java.lang.String file_description,
              java.lang.String file_size_formatted)
```

Constructeur du model pour le stockage d'un model de fichier

Parameters:

PKNoFile - Clé primaire du fichier
FKNoFolder - Clé étrangère du dossier
file_name - Nom du fichier
file_rename - Nom du fichier spécifié par l'utilisateur
file_size - Taille du fichier
file_extension - Extension du fichier
file_description - Description du fichier
file_size_formatted - Taille du fichier formattée

Methods

getFKNoFolder

`public int` **`getFKNoFolder`**`()`

Obtention de la clé étrangère vers le dossier contenant le fichier

Returns:

Clé étrangère vers le dossier

---

getFile_description

`public java.lang.String` **`getFile_description`**`()`

Obtention de la description du fichier

Returns:

Description du fichier

---

getFile_extension

`public java.lang.String` **`getFile_extension`**`()`

Obtention de l'extension du fichier

Returns:

Extension du fichier

---

getFile_name

`public java.lang.String` **`getFile_name`**`()`

Obtention du nom du fichier tel qu'il est enregistré sur le disque

Returns:

Nom du fichier sur le disque

---

getFile_rename

`public java.lang.String` **`getFile_rename`**`()`

Obtention du nom de fichier spécifié par l'utilisateur

Returns:

Nom du fichier spécifié par l'utilisateur

---

getFile_size

`public long` **`getFile_size`**`()`

Obtention de la taille du fichier

Returns:

Taille du fichier (Octets)

---

getFile_size_formatted

`public java.lang.String` **`getFile_size_formatted`**`()`

Obtention de la taille du fichier formattée

Returns:

Taille formattée

---

getPKNoFile

`public int` **`getPKNoFile`**`()`

Obtention de la clé primaire du fichier

Returns:

Clé primaire du fichier

---

setFKNoFolder

`public void` **`setFKNoFolder`**`(int fKNoFolder)`

Paramétrage de la clé étrangère

Parameters:

fKNoFolder - Clé étrangère vers le dossier

---

setFile_description

`public void` **`setFile_description`**`(java.lang.String file_description)`

Réglage de la description du fichier

Parameters:

file_description - Description du fichier

---

setFile_extension

```
public void setFile_extension(java.lang.String file_extension)
```

Réglage de l'extension du fichier

Parameters:

file_extension - Extension du fichier

---

setFile_name

```
public void setFile_name(java.lang.String file_name)
```

Réglage du nom de fichier

Parameters:

file_name - Nom du fichier

---

setFile_rename

```
public void setFile_rename(java.lang.String file_rename)
```

Réglage du nom du fichier spécifié par l'utilisateur

Parameters:

file_rename - Nom du fichier spécifié par l'utilisateur

---

setFile_size

```
public void setFile_size(long file_size)
```

Réglage de la taille du fichier

Parameters:

file_size - Taille du fichier en octets

---

setFile_size_formatted

```
public void setFile_size_formatted(java.lang.String file_size_formatted)
```

Réglage de la taille formattée du fichier

Parameters:

file_size_formatted - Taille formattée du fichier

---

setPKNoFile

```
public void setPKNoFile(int pKNoFile)
```

Fixation de la clé primaire du fichier

Parameters:

pKNoFile - Clé primaire du fichier

---

model

Class Folder

```
java.lang.Object
    |
    +--model.Folder
```

< Constructors > < Methods >

---

public class **Folder**
extends java.lang.Object

Modèle d'un dossier de transfert (Table trans_folders).

Permet le stockage des informations sur un dossier dans un objet.

Author:

Dominique Roduit

Constructors

Folder

```
public  Folder()
```

Folder

```
public  Folder(int PKNoFolder,
               int FKNoUser,
               java.lang.String folder_name,
               java.lang.String description,
               java.lang.String folder_creation_date,
               java.lang.String folder_expiration,
               boolean folder_archive)
```

Constructeur du model pour le stockage d'un model de dossier

Parameters:

PKNoFolder - Clé primaire du dossier
FKNoUser - Clé étrangère vers l'utilisateur
folder_name - Nom du dossier
folder_creation_date - Date de création du dossier
folder_expiration - Date d'expiration du dossier
folder_archive - true si le dossier est une archive

Methods

getArchive

```
public boolean getArchive()
```

Indique si le dossier est une archive ou non

Returns:

true si le dossier est une archive (expiré)

---

getCreation_date

```
public java.lang.String getCreation_date()
```

Obtention de la date de création du dossier

Returns:

Date de création du dossier

---

getDescription

```
public java.lang.String getDescription()
```

Returns:

La Description du dossier

---

getExpiration

```
public double getExpiration()
```

Obtention de la durée de vie du dossier

Returns:

Durée de vie du dossier

---

getExpiration_date

```
public java.lang.String getExpiration_date()
```

Obtention de la date d'expiration du dossier

Returns:

Date d'expiration du dossier au format MySQL

---

getFKNoUser

```
public int getFKNoUser()
```

Obtention de la clé étrangère de l'utilisateur

Returns:

Clé étrangère de l'utilisateur

---

getName

```
public java.lang.String getName()
```

Obtention du nom du dossier

Returns:

Nom du dossier

---

getPKNoFolder

```
public int getPKNoFolder()
```

Obtention de la clé primaire du dossier

Returns:

Clé primaire du dossier

---

setArchive

`public void setArchive(boolean archive)`

Définit si le dossier est une archive ou non

Parameters:

archive - true si le dossier est une archive

---

setCreation_date

`public void setCreation_date(java.lang.String creation_date)`

Définition de la date de création du dossier

Parameters:

creation_date - Date de création du dossier

---

setDescription

`public void setDescription(java.lang.String description)`

Parameters:

description - Description du dossier

---

setExpiration

`public void setExpiration(double expiration)`

Définition de la durée de vie du dossier

Parameters:

expiration - Durée de vie du dossier en jours

---

setExpiration_date

`public void setExpiration_date(java.lang.String expiration_date)`

Définition de la date d'expiration du dossier

Parameters:

expiration_date - Date d'expiration du dossier au format MySQL

---

setFKNoUser

`public void setFKNoUser(int fKNoUser)`

Réglage de la clé étrangère de l'utilisateur

Parameters:

fKNoUser - Clé étrangère de l'utilisateur

---

setName

`public void setName(java.lang.String name)`

Réglage du nom du dossier

Parameters:

name - Nom du dossier

---

setPKNoFolder

`public void setPKNoFolder(int pKNoFolder)`

Réglage de la clé primaire du dossier

Parameters:

pKNoFolder - Clé primaire du dossier

---

model

Class JournalCon

```
java.lang.Object
    |
    +--model.JournalCon
```

< Constructors > < Methods >

---

public class **JournalCon**
extends java.lang.Object

Modèle d'une action de connexion journalisée (Table trans_journal_connections).

Permet le stockage des actions de connexion dans un objet Java.

Author:

Dominique Roduit

Constructors

JournalCon

```
public  JournalCon(int pKNoJourConnection,
                   java.lang.String joco_mail,
                   java.lang.String joco_date,
                   java.lang.String joco_ip,
                   java.lang.String joco_os,
                   java.lang.String joco_browser,
                   java.lang.String joco_action,
                   java.lang.String joco_comment)
```

Methods

getJoco_action

```
public java.lang.String getJoco_action()
```

Returns:

Action déclenchée

---

getJoco_browser

```
public java.lang.String getJoco_browser()
```

Returns:

Nom de navigateur et version de la personne qui déclenche l'action

---

getJoco_comment

```
public java.lang.String getJoco_comment()
```

Returns:

Commentaire sur l'action

---

getJoco_date

```
public java.lang.String getJoco_date()
```

Returns:

La date à laquelle l'action est survenue

---

getJoco_ip

```
public java.lang.String getJoco_ip()
```

Returns:

L'adresse IP publique de la personne qui a déclenché l'action

---

getJoco_mail

```
public java.lang.String getJoco_mail()
```

Returns:

L'adresse e-mail de la personne qui déclenche l'action

---

getJoco_os

```
public java.lang.String getJoco_os()
```

Returns:

Système d'exploitation de la personne qui déclenche l'action

---

getPKNoJourConnection

```
public int getPKNoJourConnection()
```

Returns:

Clé primaire de l'action

---

setJoco_action

```
public void setJoco_action(java.lang.String joco_action)
```

Parameters:

joco_action - Action déclenchée

---

setJoco_browser

```
public void setJoco_browser(java.lang.String joco_browser)
```

Parameters:

joco_browser - Nom de navigateur et version de la personne qui déclenche l'action

---

setJoco_comment

```
public void setJoco_comment(java.lang.String joco_comment)
```

Parameters:

joco_comment - Commentaire sur l'action

---

setJoco_date

```
public void setJoco_date(java.lang.String joco_date)
```

Parameters:

joco_date - La date à laquelle l'action est survenue

---

setJoco_ip

```
public void setJoco_ip(java.lang.String joco_ip)
```

Parameters:

joco_ip - L'adresse IP publique de la personne qui a déclenché l'action

---

setJoco_mail

```
public void setJoco_mail(java.lang.String joco_mail)
```

Parameters:

joco_mail - L'adresse e-mail de la personne qui déclenche l'action

---

setJoco_os

```
public void setJoco_os(java.lang.String joco_os)
```

Parameters:

joco_os - Système d'exploitation de la personne qui déclenche l'action

---

setPKNoJourConnection

```
public void setPKNoJourConnection(int pKNoJourConnection)
```

Parameters:

pKNoJourConnection - Clé primaire de l'action

---

model

Class User

```
java.lang.Object
    |
    +--model.User
```

< Constructors > < Methods >

public class **User**
extends java.lang.Object

Modèle d'un utilisateur (Table trans_users).

Permet le stockage des informations sur un utilisateur dans un objet.

Author:

Dominique Roduit

Constructors

User

```
public  User()
```

User

```
public  User(int PKNoUser,
             java.lang.String user_mail,
             java.lang.String user_pass,
             boolean user_internal,
             boolean user_validation,
             java.lang.String user_validation_date,
             java.lang.String user_langue,
             boolean user_admin)
```

Création d'un objet, model d'utilisateur

Parameters:

PKNoUser - Clé primaire de l'utilisateur
user_mail - Adresse e-mail de l'utilisateur
user_pass - Mot de passe de l'utilisateur
user_internal - Indique si l'utilisateur est interne ou pas
user_validation - Indique si l'utilisateur a validé son compte
user_validation_date - Date de validation du compte utilisateur
user_langue - Langue de l'utilisateur
user_admin - Indique si l'utilisateur est un administrateur

Methods

getPKNoUser

```
public int getPKNoUser()
```

Obtention de la clé primaire

Returns:

Clé primaire

---

getUser_langue

```
public java.lang.String getUser_langue()
```

Retourne la langue préférée par l'utilisateur

Par défaut, la langue préférée est celle que l'utilisateur utilise dans son navigateur.

Returns:

Langue de l'utilisateur

---

getUser_mail

```
public java.lang.String getUser_mail()
```

Obtention de l'adresse e-mail de l'utilisateur

Returns:

Adresse e-mail de l'utilisateur

---

getUser_pass

```
public java.lang.String getUser_pass()
```

Obtention du mot de passe crypté de l'utilisateur

Returns:

Mot de passe crypté

---

getUser_validation

```
public boolean getUser_validation()
```

Retourne l'état du compte de l'utilisateur

Returns:

1 : Compte validé

0 : Compte non-validé

---

getUser_validation_date

```
public java.lang.String getUser_validation_date()
```

Retourne la date de validation du compte utilisateur

Returns:

Date de validation

---

isInternal

```
public boolean isInternal()
```

Obtention de l'état interne ou non de l'utilisateur

Returns:

1 : Utilisateur interne

0 : Utilisateur externe

isUser_admin

`public boolean isUser_admin()`

Returns:

Indique si l'utilisateur est un administrateur

setInternal

`public void setInternal(boolean user_internal)`

Sauvegarde le fait que l'utilisateur soit un interne

Parameters:

user_internal - 1 : Utilisateur interne

0: Utilisateur externe

setPKNoUser

`public void setPKNoUser(int PKNoUser)`

Enregistrement de la clé primaire de l'utilisateur

Parameters:

PKNoUser - Clé primaire de l'utilisateur

setUser_admin

`public void setUser_admin(boolean user_admin)`

Parameters:

D#nit - l'utilisateur comme administrateur ou non

setUser_langue

`public void setUser_langue(java.lang.String user_langue)`

Stocke la langue préférée de l'utilisateur

Parameters:

user_langue - Langue de l'utilisateur

setUser_mail

`public void setUser_mail(java.lang.String user_mail)`

Réglage de l'adresse e-mail de l'utilisateur

Parameters:

user_mail -

setUser_pass

`public void setUser_pass(java.lang.String pass)`

Définition du mot de passe de l'utilisateur

Parameters:

pass - Mot de passe de l'utilisateur (crypté)

setUser_validation

`public void setUser_validation(boolean user_validation)`

Sauvegarde le fait que l'utilisateur ait validé son compte ou non

Parameters:

user_validation - 1 : Compte validé

0 : Compte non-validé

setUser_validation_date

`public void setUser_validation_date(java.lang.String user_validation_date)`

Stocke la date de validation du compte utilisateur

Parameters:

user_validation_date - Date de validation

Package modules

Class Summary

AboutModule

Page d'affichage des informations concernant l'application.

ConditionsModule

Page d'affichage des conditions d'utilisation de l'application.

ContactModule

Page de gestion des contacts.

DownloadModule

Page de téléchargement des fichiers d'un dossier dans lequel un ou plusieurs contacts ont été invités.

FolderModule

Page d'affichage des fichiers et informations d'un dossier.

GetLinkModule

Module qui permet la récupération d'un lien de fichier ou d'un lien d'une archive de dossier.

ReMailLinkModule

Redistribution des e-mail pour l'accès à un dossier.

TransfertModule

Page de gestion des transferts.

---

modules

Class AboutModule

```
java.lang.Object
    |
    +--modules.AboutModule
```

< Constructors > < Methods >

---

public class **AboutModule**
extends java.lang.Object

Page d'affichage des informations concernant l'application.

La page inclut un bouton de retour au dernier module actif.

Les informations sont contenues dans un bloc et illustrées par le logo de la Caisse de compensation

Author:

Dominique Roduit

Constructors

AboutModule

```
public  AboutModule()
```

Methods

getBodyContent

```
public static VerticalLayout getBodyContent()
```

Insertion du contenu dans le corps de la page.

Le contenu comprend le texte, extrait de la base de données ainsi qu'une image illustrative.

---

getBodyRibbonContent

```
public static HorizontalLayout getBodyRibbonContent()
```

Chargement du bouton de retour et du titre dans le ruban du corps de la page

---

modules

Class ConditionsModule

```
java.lang.Object
    |
    +--modules.ConditionsModule
```

< Constructors > < Methods >

---

public class **ConditionsModule**
extends java.lang.Object

Page d'affichage des conditions d'utilisation de l'application.

La page inclut un bouton de retour au dernier module actif.

Les conditions sont contenues dans un bloc et illustrées par une image libre au format PNG.

Author:

Dominique Roduit

Constructors

ConditionsModule

```
public  ConditionsModule()
```

Methods

getBodyContent

```
public static VerticalLayout getBodyContent()
```

Insertion du contenu dans le corps de la page.

Le contenu comprend le texte, extrait de la base de données ainsi qu'une image illustrative.

---

getBodyRibbonContent

```
public static HorizontalLayout getBodyRibbonContent()
```

Chargement du bouton de retour et du titre dans le ruban du corps de la page

---

modules

Class ContactModule

```
java.lang.Object
    |
    +--modules.ContactModule
```

< Constructors > < Methods >

public class **ContactModule**
extends java.lang.Object

Page de gestion des contacts.

L'utilisateur peut ajouter/éditer/supprimer des contacts. Les actions effectuées sur cette page sont répercutées sur le tableau du menu affichant le même résultat

Author:

Dominique Roduit

Constructors

ContactModule

```
public  ContactModule()
```

Methods

getBodyContent

```
public static ContactTable getBodyContent()
```

Insertion du contenu dans le corps de la page.

Le contenu comprend un tableau répertoriant les contact que l'utilisateur à ajouté.

Le tableau implémente un menu contextuel qui permet la suppression et l'édition d'un contact

---

getBodyRibbonContent

```
public static HorizontalLayout getBodyRibbonContent()
```

Chargement des boutons dans le ruban du corps de page

---

modules

Class DownloadModule

```
java.lang.Object
    |
    +--modules.DownloadModule
```

< Constructors > < Methods >

public class **DownloadModule**
extends java.lang.Object

Page de téléchargement des fichiers d'un dossier dans lequel un ou plusieurs contacts ont été invités.

Author:

Dominique Roduit

2013-03-08

Constructors

DownloadModule

```
public  DownloadModule(int PKNoFolder,
                       int PKNoContact,
                       java.lang.String mail)
```

Création d'une page de téléchargement des fichiers d'un dossier spécifique

Parameters:

PKNoFolder - Clé primaire du dossier consulté
PKNoContact - Clé primaire du contact invité au dossier
mail - Adresse e-mail du contact invité au dossier

Methods

getBodyContent

```
public Table getBodyContent()
```

Insertion du contenu dans le corps de la page.

Le contenu comprend un tableau listant les fichiers du dossier ainsi que les informations y relatives.

---

getBodyRibbonContent

```
public HorizontalLayout getBodyRibbonContent()
```

Chargement des boutons dans le ruban du corps de la page

---

modules

Class FolderModule

```
java.lang.Object
    |
    +--modules.FolderModule
```

< Fields > < Constructors > < Methods >

---

public class **FolderModule**
extends java.lang.Object

Page d'affichage des fichiers et informations d'un dossier.

Le ruban n'est pas rechargé par cette class, le bouton d'ajout d'un dossier est donc conservé.

Un panneau affiche les informations du dossier

- Création
- Expiration
- Disponibilité
- Destinataires
- Statut (Archivé ou non)
- ...

Les types de fichiers sont illustrés par une icône d'extension.

Author:

Dominique Roduit

Fields

ACTION_GET_LINK

```
protected static Action ACTION_GET_LINK
```

Action pour la récupération du lien de téléchargement

Constructors

FolderModule

```
public  FolderModule()
```

Methods

getBodyContent

```
public static HorizontalLayout getBodyContent(int PKNoFolder)
```

Insertion du contenu dans le corps de page.

Le contenu comprend le tableau répertoriant les fichiers ainsi que les informations relatives au dossier.

---

getBodyRibbonContent

```
public static Button getBodyRibbonContent()
```

Chargement des boutons dans le ruban du corps de page

modules

Class GetLinkModule

```
java.lang.Object
    |
    +--CustomComponent
        |
        +--modules.GetLinkModule
```

< Constructors >

public class **GetLinkModule**
extends CustomComponent

Module qui permet la récupération d'un lien de fichier ou d'un lien d'une archive de dossier.

L'utilisateur peut ainsi télécharger lui même ses propres fichiers ou partager un fichier avec n'importe qui d'autre en lui donnant uniquement l'URL vers le fichier.

Author:
Dominique Roduit

Constructors

GetLinkModule

```
public  GetLinkModule(java.lang.String filename)
```

Affichage des composants (TextArea, boutons)

Parameters:

filename - Nom du fichier sur le disque

modules

Class ReMailLinkModule

```
java.lang.Object
    |
    +--CustomComponent
        |
        +--modules.ReMailLinkModule
```

< Constructors >

public class **ReMailLinkModule**
extends CustomComponent

Redistribution des e-mail pour l'accès à un dossier.

Ce module peut être utile si un des contacts dit ne pas avoir reçu le mail pour X raisons, il permet de réenvoyer le lien crypté.

Par contre la date d'expiration est conservée et ne change pas.

Author:
Dominique Roduit

Constructors

ReMailLinkModule

```
public  ReMailLinkModule(int pKNoFolder)
```

Affichage des composants du module (Tableau, boutons, textes, ...)

Parameters:

pKNoFolder - Clé primaire du dossier concerné

modules

Class TransfertModule

```
java.lang.Object
    |
    +--modules.TransfertModule
```

< Constructors > < Methods >

public class **TransfertModule**
extends java.lang.Object

Page de gestion des transferts.

Elle affiche un bouton dans le ruban du menu pour permettre la création d'un nouveau dossier.

La fenêtre d'assistance à la navigation est implémentée dans cette class.

Lorsqu'un dossier est affiché, c'est la class {@link FolderModule} qui est implémentée.

Author:
Dominique Roduit

Constructors

TransfertModule

```
public  TransfertModule()
```

Methods

getBodyContent

`public static VerticalLayout` **`getBodyContent`**`()`

Insertion du contenu de la page dans le corps de l'application.

Contenu :

- Assistance à la première utilisation
- Page de démarrage

---

getBodyRibbonContent

`public static Button` **`getBodyRibbonContent`**`()`

Chargement des boutons dans le ruban du wrapper principal.

Bouton "Nouveau dossier"

Package sql.query

Class Summary

tbl_contacts

Cette class contient toutes les requêtes SQL utiles à la gestion de la table des contacts.

tbl_files

Contient toutes les requêtes effectuées sur la table **trans_files**

tbl_folders

Contient toutes les requêtes effectuées sur la table **trans_folders**

tbl_journal_con

JOURNALISATION

Contient toutes les requêtes effectuées sur la table **trans_journal_connections**

tbl_journal_fold

JOURNALISATION

Contient toutes les requêtes effectuées sur la table **trans_journal_download**

tbl_recipients

Contient toutes les requêtes effectuées sur la table **trans_recipients**

tbl_users

Contient toutes les requêtes effectuées sur la table **trans_users**

---

sql.query

Class tbl_contacts

```
java.lang.Object
    |
    +--sql.query.tbl_contacts
```

< Constructors > < Methods >

public class **tbl_contacts**
extends java.lang.Object

Cette class contient toutes les requêtes SQL utiles à la gestion de la table des contacts.

Author:

Dominique Roduit

Constructors

tbl_contacts

```
public  tbl_contacts()
```

Methods

getContactByCryptedURL

```
public static java.lang.String getContactByCryptedURL(java.lang.String
URLParameter)
```

Sélection d'un contact en fonction d'un paramètre crypté passé dans l'URL

Parameters:

URLParameter - Valeur du paramètre de l'URL

Returns:

Requête SQL

---

getDeleteContactQuery

```
public static java.lang.String getDeleteContactQuery(java.lang.String PK)
```

Requête pour la suppression d'un contact par sa clé primaire

Parameters:

PK - (String) Clé primaire du contact

Returns:

(String) Requête SQL Formatée

---

getInsertContactQuery

```
public static java.lang.String getInsertContactQuery(java.lang.String name,
                                                      java.lang.String email)
```

Requête pour l'insertion d'un contact

Parameters:

name - (String) Nom du contact
email - (String) E-mail du contact

Returns:

(String) Requête SQL Formatée

---

getLastContact

```
public static java.lang.String getLastContact()
```

Retourne l'ID du dernier contact

Returns:

ID du dernier contact

---

getNumberOfContact

```
public static int getNumberOfContact()
```

Retourne le nombre de contact dans la liste d'un utilisateur

Returns:

Requête SQL

---

getSelectAllContactQuery

```
public static java.lang.String getSelectAllContactQuery()
```

Requête pour la sélection de la liste des contacts

Returns:

(String) Requête SQL Formatée

---

getSelectByPKContactQuery

```
public static java.lang.String getSelectByPKContactQuery(java.lang.String PK)
```

Requête pour la sélection d'un contact par sa clé primaire

Parameters:

PK - (String) Clé primaire du contact

Returns:

(String) Requête SQL Formatée

---

getSelectContactsFromFolder

```
public static java.lang.String getSelectContactsFromFolder(int PKNoFolder)
```

Sélection des contacts attribués comme destinataires d'un dossier

Parameters:

PKNoFolder - Clé primaire du dossier

Returns:

Requête SQL

---

getUpdateContactQuery

```
public java.lang.String getUpdateContactQuery(java.lang.String PK,
                                               java.lang.String name,
                                               java.lang.String email)
```

Requête pour la mise à jour d'un contact

Parameters:

PK - (String) Clé primaire du contact
name - (String) Nom du contact
email - (String) E-mail du contact

Returns:

(String) Requête SQL Formatée

---

sql.query

Class tbl_files

```
java.lang.Object
    |
    +--sql.query.tbl_files
```

< Constructors > < Methods >

public class **tbl_files**
extends java.lang.Object

Contient toutes les requêtes effectuées sur la table **trans_files**

Author:

Dominique Roduit

Constructors

tbl_files

```
public  tbl_files()
```

Methods

getFileNumberFromFolder

```
public static int getFileNumberFromFolder(int PKNoFolder)
```

Retourne le nombres de fichiers contenus dans un dossier

Parameters:

FKNoFolder - Clé Primaire du dossier

Returns:

Nombre de fichiers du dossier

---

getFileSizeFromFolder

```
public static int getFileSizeFromFolder(int PKNoFolder)
```

Retourne la taille total des fichiers contenus dans un dossier

Parameters:

FKNoFolder - Clé Primaire du dossier

Returns:

Taille total des fichiers contenu dans le dossier (en octet)

---

getSelectAllFilesFromFolder

```
public static java.lang.String getSelectAllFilesFromFolder(int FKNoFolder)
```

Retourne tous les fichiers d'un dossier

Parameters:

FKNoFolder - La clé primaire du dossier contenant les fichiers

Returns:

Requête SQL

sql.query

Class tbl_folders

```
java.lang.Object
    |
    +--sql.query.tbl_folders
```

< Constructors > < Methods >

public class **tbl_folders**
extends java.lang.Object

Contient toutes les requêtes effectuées sur la table **trans_folders**

Author:

Dominique Roduit

Constructors

tbl_folders

```
public  tbl_folders()
```

Methods

getAllFolders

```
public static java.lang.String getAllFolders()
```

Requête pour la sélection de la liste des contacts

Returns:

(String) Requête SQL Formatée

getDeleteFolder

```
public static java.lang.String getDeleteFolder(java.lang.String PK)
```

Requête pour la suppression d'un dossier

Parameters:

PK - La clé primaire du dossier à supprimer

Returns:

Requête SQL Formatée

getFolderByCryptedURL

```
public static java.lang.String getFolderByCryptedURL(java.lang.String
URLParameter)
```

Sélection d'un dossier en fonction d'un paramètre crypté passé dans l'URL

Parameters:

URLParameter - Valeur du paramètre de l'URL

Returns:

Requête SQL

getInsertFolder

```
public static java.lang.String getInsertFolder(java.lang.String name,
                                                double expiration,
                                                java.lang.String description)
```

Requête pour l'insertion d'un dossier

Parameters:

name - Nom du dossier
expiration - Nombre de jour ou nous mettons a disposition le dossier

Returns:

Requête SQL

getNumberOfDayFolderIsAvailable

```
public static int getNumberOfDayFolderIsAvailable(int PKNoFolder)
```

Retourne le nombre de jour ou le dossier est disponible (Date d'expiration-Date de création)

Parameters:

PKNoFolder - Clé primaire du dossier

Returns:

Nombre de jour de disponibilité du dossier

getNumberOfFolders

```
public static int getNumberOfFolders()
```

Retourne le nombre de dossiers d'un utilisateur

Returns:

Requête SQL

getRemainingDaysFolderIsAvailable

```
public static int getRemainingDaysFolderIsAvailable(int PKNoFolder)
```

Retourne le nombre de jours restant pour un dossier avant son expiration

Parameters:

PKNoFolder - Clé primaire du dossier

Returns:

Durée de vie restante du dossier

getRemainingHoursFolderIsAvailable

```
public static int getRemainingHoursFolderIsAvailable(int PKNoFolder)
```

Retourne le nombde d'heures restantes avant l'expiration d'un dossier

Parameters:

PKNoFolder - Clé primaire du dossier

Returns:

Heures restantes avant la destruction du dossier

getSelectFolderInfos

```
public static java.lang.String getSelectFolderInfos(int pKNoFolder)
```

Retourne les informations sur un dossier

Parameters:

pKNoFolder - Clé primaire du dossier

Returns:

Requête SQL

getSelectMaxPK

```
public static java.lang.String getSelectMaxPK()
```

Requête pour la récupération du dernier dossier créé par l'utilisateur

Returns:

La PK du dernier dossier créé par l'utilisateur

sql.query

Class tbl_journal_con

```
java.lang.Object
    |
    +--sql.query.tbl_journal_con
```

< Constructors > < Methods >

public class **tbl_journal_con**
extends java.lang.Object

JOURNALISATION

Contient toutes les requêtes effectuées sur la table **trans_journal_connections**

Author:

Dominique Roduit

Constructors

tbl_journal_con

```
public  tbl_journal_con()
```

Methods

getInsertQuery

```
public static java.lang.String getInsertQuery(java.lang.String email,
                                               java.lang.String action,
                                               java.lang.String comment)
```

Insertion d'une entrée dans le journal des connexions

Parameters:

email - E-mail de l'utilisateur
action - Action effectuée / Evenement
comment - Commentaire sur l'action effectuée

Returns:

Requête SQL formatée

getNumberOfConnexionsWhithoutValidation

```
public static int getNumberOfConnexionsWhithoutValidation(java.lang.String
email)
```

Retourne le nombres de fois qu'un utilisateur a tenté de se connecter sans valider son compte

Parameters:

email - Adresse e-mail de l'utilisateur

Returns:

Nombre de connection sans avoir validé le compte

---

getSelectConnexions

```
public static java.lang.String getSelectConnexions(java.lang.String where)
```

Requête pour la sélection des journalisations sur les connexions

Returns:

Requête SQL

---

getSelectValidations

```
public static java.lang.String getSelectValidations(java.lang.String where)
```

Requête pour la sélection des journalisations sur les validations des comptes

Returns:

Requête SQL

---

sql.query

Class tbl_journal_fold

```
java.lang.Object
    |
    +--sql.query.tbl_journal_fold
```

< Constructors > < Methods >

---

public class **tbl_journal_fold**
extends java.lang.Object

JOURNALISATION

Contient toutes les requêtes effectuées sur la table **trans_journal_download**

Author:

Dominique Roduit

Constructors

tbl_journal_fold

```
public  tbl_journal_fold()
```

Methods

getInsertQuery

```
public static java.lang.String getInsertQuery(java.lang.String action,
                                               int FKNoContact,
                                               int FKNoFolder,
                                               int FKNoFile)
```

Insertion d'une entrée dans le journal des téléchargements

Parameters:

FKNoContact - Clé étrangère vers le contact
FKNoFolder - Clé étrangère vers le dossier
FKNoFile - Clé étrangère vers le fichier

Returns:

Requête SQL formatée

---

getSelectAll

```
public static java.lang.String getSelectAll(java.lang.String where)
```

Requête de sélection de toutes les informations journalisées sur les dossiers

Parameters:

where - Clause WHERE de la requête

Returns:

Requête SQL

---

getSelectAllForContact

```
public static java.lang.String getSelectAllForContact(int pKNoContact)
```

Retourne la requête de sélection des informations journalisées pour un contact précisé

Parameters:

pKNoContact - Clé primaire du contact

Returns:

Requête SQL

---

getSelectAllForContactAndFolder

```
public static java.lang.String getSelectAllForContactAndFolder(int
pKNoContact,
                                                               int PKNoFolder)
```

Retourne la requête de sélection des informations journalisées sur un dossier, pour un contact précisé

Parameters:

pKNoContact - Clé primaire du contact
PKNoFolder - Clé primaire du dossier

Returns:

Requête SQL

---

sql.query

Class tbl_recipients

```
java.lang.Object
    |
    +--sql.query.tbl_recipients
```

< Constructors > < Methods >

---

public class **tbl_recipients**
extends java.lang.Object

Contient toutes les requêtes effectuées sur la table **trans_recipients**

Author:

Dominique Roduit

Constructors

tbl_recipients

```
public  tbl_recipients()
```

Methods

getInsertRecipient

```
public static java.lang.String getInsertRecipient(int FKNoFolder,
                                                   int FKNoContact)
```

Requête d'insertion pour les destinataires

---

getSelectRecipientsFromFolder

```
public static java.lang.String getSelectRecipientsFromFolder(int pKNoFolder)
```

Requête de sélection des destinataire pour un dossier

Parameters:

pKNoFolder - PK du dossier

Returns:

Requête SQL

---

sql.query

Class tbl_users

```
java.lang.Object
    |
    +--sql.query.tbl_users
```

< Constructors > < Methods >

---

public class **tbl_users**
extends java.lang.Object

Contient toutes les requêtes effectuées sur la table **trans_users**

Author:

Dominique Roduit

Constructors

tbl_users

```
public  tbl_users()
```

Methods

getDeleteUser

```
public static java.lang.String getDeleteUser(int pkNoUser)
```

Requête de suppression d'un utilisateur

Parameters:

pkNoUser - Clé primaire de l'utilisateur

Returns:

Requête SQL

---

getInsertQuery

```
public static java.lang.String getInsertQuery(java.lang.String email,
                                               java.lang.String pass,
                                               boolean isInternal)
```

Retourne la requête d'insertion d'un utilisateur

Parameters:

email - Adresse e-mail
pass - Mot de passe
internal - L'utilisateur est un interne ou non

Returns:

Requête SQL

---

getSelectAll

```
public static java.lang.String getSelectAll()
```

Sélection de tous les contacts de l'interface utilisateur

Returns:

Requête SQL

---

getSelectUserInfos

```
public static java.lang.String getSelectUserInfos(int PKNoUser)
```

Requête pour la sélection des informations sur un utilisateur

Parameters:

PKNoUser - Clé primaire de l'utilisateur

Returns:

Requête SQL

---

getUpdByMailValidation

```
public static java.lang.String getUpdByMailValidation(java.lang.String email)
```

Mise à jour du champs de validation du compte utilisateur

Parameters:

email - Adresse de l'utilisateur qui valide son compte

Returns:

Requête SQL

---

getUpdForAdminAssign

```
public static java.lang.String getUpdForAdminAssign(int PKNoUser,
                                                    boolean newValue)
```

Requête de mise à jour d'un utilisateur (Admin ou pas)

Parameters:

newValue - true si admin

Returns:

Requête SQL

---

getUpdForValidation

```
public static java.lang.String getUpdForValidation(int PKNoUser,
                                                   boolean newValue)
```

Requête de validation d'un compte utilisateur

Parameters:

newValue - true pour valider

Returns:

Requête SQL

getUpdLanguage

`public static java.lang.String` **`getUpdLanguage`**`(java.lang.String newLanguage)`

Requête pour la mise à jour de la langue de l'utilisateur

Parameters:

newLanguage - Langue choisie par l'utilisateur

Returns:

Requête SQL

getUserByCryptedURL

`public static java.lang.String` **`getUserByCryptedURL`**`(java.lang.String URLParameter)`

Sélection d'un utilisateur en fonction d'un paramètre crypté passé dans l'URL

Parameters:

URLParameter - Valeur du paramètre de l'URL

Returns:

Requête SQL

getUserByEmail

`public static java.lang.String` **`getUserByEmail`**`(java.lang.String email)`

Sélection d'un utilisateur par son adresse e-mail

Parameters:

email - Adresse de l'utilisateur

Returns:

Requête SQL

Package toolbox

Class Summary

Mailer

Class qui répertorie différents outils relatifs à l'utilisation des adresses e-mails

Outils principaux :

- Envoi de mails en java via le protocole SMPT (requis : librairie javamail).

Mailer.EmailValidator

Cette classe regroupe des outils très utiles pouvant être appliqués aux adresses mails.

SHA256

Class de Hachage de valeurs.

Sql

Contient les méthodes pour la connexion et l'exécution de requêtes sur la base de données.

Utilities

Classe regroupant les fonctions importantes utilisées globalement dans toute l'application

ZipFileWritter

Création d'un fichier d'archive au format ZIP, compression et ajours de fichiers à l'archive.

toolbox

Class Mailer

```
java.lang.Object
    |
    +--toolbox.Mailer
```

< Fields > < Constructors > < Methods >

public class **Mailer**
extends java.lang.Object

Class qui répertorie différents outils relatifs à l'utilisation des adresses e-mails

Outils principaux :

- Envoi de mails en java via le protocole SMPT (requis : librairie javamail).
- Validation et contrôles de formats d'adresses

Author:

Dominique Roduit

Fields

APP_MAIL_FROM

```
public static final java.lang.String APP_MAIL_FROM
```

Adresse mail utilisée pour envoyer des messages automatiques aux utilisateurs

---

APP_MAIL_FROM_NAME

```
public static final java.lang.String APP_MAIL_FROM_NAME
```

Nom d'affichage de l'adresse e-mail définie par APP_MAIL_FROM

Constructors

Mailer

```
public  Mailer()
```

Methods

send

```
public static void send(java.lang.String to,
                        java.lang.String subject,
                        java.lang.String content)
```

Envoi d'un mail automatique, l'adresse de l'auteur reste fixe et le nom aussi

Parameters:

to - Destinataires séparés par des virgules
subject - Sujet du message
content - Contenu du mail

---

send

```
public static void send(java.lang.String from,
                        java.lang.String from_name,
                        java.lang.String to,
                        java.lang.String subject,
                        java.lang.String content)
```

Envoi d'un mail au(x) destinataire(s) spécifié(s).

Parameters:

from - (String) Auteur du mail
from_name - (String) Nom de l'auteur du mail
to - (String) Destinataires (séparés par des virgules)
subject - (String) Sujet du message
content - (String) Contenu du mail au format HTML

---

sendAccountValidationMail

```
public static void sendAccountValidationMail(java.lang.String to)
```

Envoi du mail contenant le lien de validation du compte utilisateur

Parameters:

to - Adresse e-mail de destination

---

sendDownloadLink

```
public static void sendDownloadLink(java.lang.String to,
                                    int PKNoFolder,
                                    int PKNoContact,
                                    java.lang.String expiration_date,
                                    int expiration,
                                    java.lang.String description,
                                    java.lang.String folder_name)
```

Envoi du mail contenant le lien de téléchargement d'un dossier. Il peut être envoyé à un ou plusieurs destinataires.

Parameters:

to - Un ou plusieurs destinataires (ex. dominique@roduit.com, dominique.roduit@avs.vs.ch)
PKNoFolder - Clé primaire du dossier envoyé
PKNoContact - Clé primaire du contact à qui le dossier est envoyé
expiration_date - Date d'expiration du dossier
expiration - Durée de vie du dossier en jours

---

toolbox

Class Mailer.EmailValidator

```
java.lang.Object
    |
    +--toolbox.Mailer.EmailValidator
```

< Constructors > < Methods >

---

public static class **Mailer.EmailValidator**
extends java.lang.Object

Cette classe regroupe des outils très utiles pouvant être appliqués aux adresses mails. Elle peut être instanciée comme suit : Mailer.EmailValidator emailValidator = new Mailer.EmailValidator();

Author:

Dominique Roduit

Constructors

Mailer.EmailValidator

```
public  Mailer.EmailValidator()
```

Methods

is_internal

```
public static boolean is_internal(java.lang.String email)
```

Contrôle que la personne soit interne (de la cccvs) ou non

Returns:

true : personne interne (de la cccvs)

false : personne externe

---

validate

```
public static boolean validate(java.lang.String mailAddresse)
```

Validation du format d'une adresse e-mail

Parameters:

mailAddresse - Adresse e-mail à valider

Returns:

true : Adresse valide

false : Adresse invalide

---

toolbox

Class SHA256

```
java.lang.Object
    |
    +--toolbox.SHA256
```

< Constructors > < Methods >

public class **SHA256**
extends java.lang.Object

Class de Hachage de valeurs.

Cette class permet le hachage de valeur en utilisant l'algorithme de cryptage **SHA-256**.

Le hachage est particulièrement utilisé pour le cryptage de paramètres passés en paramètres de l'URL

Constructors

SHA256

```
public  SHA256()
```

Methods

getHashValue

```
public static java.lang.String getHashValue(java.lang.String str)
```

Hashage d'une chaine de caractère en SHA-256

Parameters:

str - Chaine de caractère à crypter

Returns:

La chaîne cryptée

---

toolbox

Class Sql

```
java.lang.Object
    |
    +--toolbox.Sql
```

All Implemented Interfaces:

java.io.Serializable

< Methods >

---

public class **Sql**
extends java.lang.Object
implements java.io.Serializable

Contient les méthodes pour la connexion et l'exécution de requêtes sur la base de données.

Les méthodes les plus importantes sont les suivantes :

- **query** : Exécution d'une requête SQL de type SELECT, que l'on peut récupérer dans un objet ResultSet.
- **exec** : Exécution d'une requête ne retournant aucun résultat, de type INSERT, UPDATE, DELETE
- **Disconnect** : Destruction de la connexion active pour la libération des ressources

Methods

Connect

```
public java.sql.Connection Connect()
```

Créer la connexion à la base de donnée.

Pour des requêtes ne nécessitant pas de liaisons de données avec des composants VAADIN

Returns:

Connexion

ConnectContainerTest

```
public SimpleJDBCConnectionPool ConnectContainerTest()
                                throws java.sql.SQLException
```

Créer la connexion à la base de donnée

Pour les requêtes qui retournent des données, tel que SELECT

Returns:

SimpleJDBCConnectionPool Pool simple de connexion JDBC

Throws:

java.sql.SQLException -

DisconectContainer

```
public void DisconectContainer(SimpleJDBCConnectionPool connectionContainer)
```

Ferme la connexion d'un container à la base de donnée

Parameters:

connectionContainer - La connexion du container à détruire

Disconnect

```
public static void Disconnect()
```

Ferme la connexion active s'il en existe une

Disconnect

```
public void Disconnect(java.sql.Connection connect)
```

Ferme la connexion passée en paramètre

Parameters:

Connection - La connexion à détruire

---

exec

```
public static void exec(java.lang.String query)
```

Exécute une requête SQL de type UPDATE, INSERT, DELETE.

Crée une connexion globale si aucune n'est déjà existante.

Parameters:

query - Requête SQL (UPDATE, INSERT, DELETE)

---

execSQL

```
public SQLContainer execSQL(SimpleJDBCConnectionPool connectionContainer,
                            java.lang.String SQLQuery)
                     throws java.sql.SQLException
```

Exécute la requête SQL et renvoie le résultat dans un SQLContainer

Parameters:

query - Requête SQL à exécuter

Returns:

(SQLContainer) Un SQLContainer contenant le résultat de la requête

Throws:

java.sql.SQLException -

---

execSQLSimple

```
public java.lang.Boolean execSQLSimple(java.sql.Connection connect,
                                       java.lang.String SQLQuery)
                        throws java.sql.SQLException
```

Exécute la requête SQL passée en paramètre

Parameters:

SQLQuery - Requête SQL à exécuter

Returns:

(Boolean) true si réussi

Throws:

java.sql.SQLException -

---

getInstance

```
public static Sql getInstance()
```

Retourne une instance de la class {@link Sql}

Returns:

Sql L'instance

---

query

```
public static java.sql.ResultSet query(java.lang.String query)
```

Exécute une requête SQL de type SELECT et retourne les résultats dans un ResultSet.

Crée une connexion globale si aucune n'existe déjà

Parameters:

query - Requête SQL (SELECT)

Returns:

(ResultSet) Résultat de la requête SELECT

---

rowCount

```
public static int rowCount()
```

Accesseur qui retourne le nombre de résultats retournés par les méthodes "query" et "exec".

Returns:

(int) Nombre de résultats de la dernière requête SQL exécutée (tout types de requêtes).

---

toolbox

Class Utilities

```
java.lang.Object
    |
    +--toolbox.Utilities
```

< Constructors > < Methods >

---

public class **Utilities**
extends java.lang.Object

Classe regroupant les fonctions importantes utilisées globalement dans toute l'application

Author:

Roduit Dominique

Constructors

Utilities

```
public  Utilities()
```

Methods

cleanFileName

```
public static java.lang.String cleanFileName(java.lang.String name)
```

Nettoye le nom d'un fichier

- Supprime les accents
- Supprime les majuscules
- Supprime les espaces
- Limite la taille de la chaine

Parameters:

name - Le nom du fichier à nettoyer

Returns:

Le nom du fichié proprement démuni de tout caractère embêtant

---

formatSQL

```
public static java.lang.String formatSQL(boolean in)
```

Formatte un boolean pour la base de données MySQL

Parameters:

in - Le boolean

Returns:

Boolean formaté (0 ou 1)

---

formatSQL

```
public static java.lang.String formatSQL(java.lang.Integer in)
```

Formate une variable pouvant être nulle pour une requête sql

Parameters:

in - Variable à formater (int)

Returns:

(String) Variable formatée

---

formatSQL

```
public static java.lang.String formatSQL(java.lang.String in)
```

Formate une variable pouvant être nulle pour une requête sql

Parameters:

in - Variable a formater (String)

Returns:

(String) Variable formatée

---

formatSQLDate

```
public static java.lang.String formatSQLDate(java.lang.String date)
```

Formate une date de la base de données MySQL. Le format d'affichage est spécifié par défaut dans la table des paramètres

Parameters:

date - Date au format MySQL

Returns:

Date Formatée

---

formatSQLDate

```
public static java.lang.String formatSQLDate(java.lang.String date,
                                             java.lang.String format)
```

Formate une date de la base de données MySQL

Parameters:

date - Date à formater (format SQL : 2013-03-11 08:16:00.456)
format - Format d'affichage de la date

- **dd** = jour
- **MM** = mois
- **MMMM** = mois en lettres
- **YY** = année
- **HH** = heure
- **mm** = minute
- **ss** = seconde

Returns:

La date formatée

---

formatSQLDateWtText

```
public static java.lang.String formatSQLDateWtText(java.lang.String date,
                                                   java.lang.String format)
```

Formatte une date du format SQL vers le format donné et affiche en texte les dates Aujourd'hui et Demain.

Parameters:

date - Date a formatter
format - Format de la date

Returns:

Date formattée

---

formatSize

```
public static java.lang.String formatSize(long size)
```

Formate la taille d'un fichier pour l'affichage

Parameters:

size - Taille du fichier en octet

Returns:

Taille formatée

---

getBrowserAndVersion

```
public static java.lang.String getBrowserAndVersion()
```

Retourne le nom du navigateur et sa version

Returns:

Nom du navigateur et sa version

---

getCurrentDate

```
public static java.lang.String getCurrentDate()
```

Récupère la date et l'heure actuel

Returns:

Date Date actuelle au format 14:49:35.769

---

getDate

```
public static java.lang.String getDate(java.lang.String format)
```

Retourne la date actuelle au format désiré

Parameters:

format - Format de date (ex. yyyyMMdd)

Returns:

Date au format désiré

---

getEmailSuffixe

```
public static java.lang.String getEmailSuffixe(java.lang.String email)
```

Retourne le suffixe d'une adresse e-mail (ex. avs.vs.ch)

Parameters:

email - Adresse e-mail

Returns:

Suffixe de l'adresse spécifiée

---

getExtension

```
public static java.lang.String getExtension(java.lang.String name)
```

Retourne l'extension d'un fichier en fonction de son nom

Parameters:

name - Le nom du fichier

Returns:

Extension du fichier

---

getFileSizeFromFormatedSize

```
public static long getFileSizeFromFormatedSize(java.lang.String size)
```

Retourne une taille en octet par rapport a une chaine de caractère formatée

Parameters:

size - Taille formatée

Returns:

Taille en octet

---

getFolderArchiveName

```
public static java.lang.String getFolderArchiveName(java.lang.String
folder_name)
```

Retourne le nom de l'archive d'un dossier par rapport à son nom

Parameters:

folder_name - Nom du dossier

Returns:

Nom de l'archive

---

getFolderArchiveName

```
public static java.lang.String getFolderArchiveName(java.lang.String
folder_name,
                                                     java.lang.String
folder_creation_date)
```

Retourne le nom de l'archive d'un dossier par rapport à son nom et sa date de création

Parameters:

folder_name - Nom du dossier
folder_creation_date - Date de création du dossier

Returns:

Nom de l'archive

---

getImgFromExtension

```
public static Embedded getImgFromExtension(java.lang.String ext)
```

Retourne le chemin vers l'image d'une extension de fichier

Parameters:

ext - Extension du fichier

Returns:

Chemin et nom de l'image

---

getInternalName

```
public static java.lang.String getInternalName(java.lang.String email)
```

Retourne le nom d'une personne interne par rapport à son adresse e-mail

Parameters:

email - Adresse e-mail de la personne

Returns:

Nom de la personne si c'est une interne

---

getNewFileName

```
public static java.lang.String getNewFileName(java.lang.String
originalFileName)
```

Renvoi le nom de fichier formatté comme nous le voulons pour l'enregistrement sur le disque.

Parameters:

originalFileName - Nom du fichier original

Returns:

Nom du fichier tel qu'il sera enregistré sur le disque

---

getOS

```
public static java.lang.String getOS()
```

Retourne le nom du système d'exploitation

Returns:

Nom du système d'exploitation

---

getURLForFile

```
public static java.lang.String getURLForFile(java.lang.String file)
```

Retourne l'URL d'un fichier

Parameters:

file - Fichier pour lequel on veux obtenir l'URL

Returns:

URL du fichier

---

getUploadDir

```
public static java.lang.String getUploadDir()
```

Retourne le dossier d'upload du serveur. Si on est en local, la méthode retourne une chaine vide.
IMPORTANT !!! - A n'utiliser que pour spécifier des URL !

Pour spécifier le chemin de destination pour l'upload d'un fichier, utiliser Global.UPLOAD_DIR

Returns:

Dossier d'upload sur le serveur (ex. /files/)

---

htmlentities

```
public static java.lang.String htmlentities(java.lang.String htmlString)
```

Suppression du code HTML dans une chaine

Parameters:

htmlString - Chaine de caractère contenant le code HTML

Returns:

Chaine parsée

---

isInternal

```
public static boolean isInternal(java.lang.String email)
```

Indique si la personne est une interne ou non

Parameters:

email - Adresse e-mail de la personne

Returns:

true si la personne est une interne

---

parmConverter

```
public static java.lang.String parmConverter(java.lang.String content)
```

Remplace tous les paramètres d'une chaine de la forme %PARAMETRE% par leur valeurs respectives

Parameters:

content - Chaine de caractère dans laquelle se trouve les paramètres

Returns:

Chaine avec les valeurs des paramètres

---

removeAccent

```
public static java.lang.String removeAccent(java.lang.String s)
```

Supprime les accentuations d'une chaine de caractères

Parameters:

arg - (String) Chaine dans laquelle supprimer les accents

Returns:

Chaine sans accents

---

zeroFill

```
public static java.lang.String zeroFill(int number)
```

Formate une unité avec un 0 devant

Parameters:

number - Nombre à formatter

Returns:

Nombre formatté

---

toolbox

Class ZipFileWritter

```
java.lang.Object
    |
    +--toolbox.ZipFileWritter
```

< Constructors > < Methods >

public class **ZipFileWritter**
extends java.lang.Object

Création d'un fichier d'archive au format ZIP, compression et ajours de fichiers à l'archive.

Author:

Fobec 2011

Constructors

ZipFileWritter

```
public  ZipFileWritter(java.lang.String zipFile)
        throws java.io.FileNotFoundException
```

Constructeurr : creation d'une nouvelle archive

Parameters:

zipFile - Nom du fichier ZIP à créer

Throws:

java.io.FileNotFoundException -

Methods

addFile

```
public void addFile(java.lang.String fileName)
            throws java.io.FileNotFoundException,
                   java.io.IOException
```

Ajoute un fichier au fichier zip

Parameters:

fileName - Chemin vers le fichier

Throws:

java.io.FileNotFoundException -
java.io.IOException -

---

close

```
public void close()
            throws java.io.IOException
```

Fermer le fichier flux de création du fichier zip

Throws:

java.io.IOException -

INDEX

F

G

H

V

W

Z

CCCVs-Transfert

Code source du projet

Caisse de compensation du canton du Valais
Ausgleichskasse Wallis

Service informatique

Dominique Roduit

27/03/2013

dominique.roduit@avs.vs.ch

PACKAGE

COMMON.COMPONENT

- AccordionMenu
- ContactTable
- MenuAdmin
- MenuContactTable
- MenuFolderTable
- PanelLight
- SQLTable
- **upload**
  - CustomUpload
  - FileInfo
  - FileUp
  - MultipleFileUpload
  - TempFileFactory2
  - *UploadActionListener*

Composants racines personnalisés qui constituent les éléments de bases de l'interface utilisateur.

Le composant de transfert de fichier occupe un sous-package distinct.

```java
package common.component;

import sql.query.tbl_contacts;

/**
 * Création du composant accordéon du menu qui contient les modules ContactTable
 * et MenuFolderTable.
 *
 * @author Dominique Roduit
 *
 */
public class AccordionMenu extends HorizontalLayout implements
        Accordion.SelectedTabChangeListener {
    /**
     * Contient toutes les instances qui doivent être accessibles dans toute
     * l'application
     **/
    private static GlobalObjects global = CccvsTransfert.getGlobalMethod();
    /** Instance de la fenêtre principale **/
    private static Window mainWindow = CccvsTransfert.mainWindow;
    /** Layout Global de l'application **/
    private static MainLayout mainLayout;
    /** Menu accordéon **/
    private static Accordion accMenu;
    /** Instance du tableau contenant les contact du menu **/
    private static MenuContactTable menuContactTable;
    /** Instance du tableau contenant les dossiers du menu **/
    private MenuFolderTable menuFolderTable;
    /** Instance du paneau d'administration **/
    private MenuAdmin menuAdmin;
    /** Texte de l'onglet du gestionnaire de fichiers **/
    private static String CAPTION_TAB_FILE_MANAGER = "";
    /** Texte de l'onglet des contacts **/
    private static String CAPTION_TAB_CONTACTS = "";
    /** Texte de l'onglet d'administration **/
    private static String CAPTION_TAB_ADMIN = "";
    /** Stocke le caption de l'onglet sélectionné **/
    private static String selectedTab = "";

    public AccordionMenu() {
        mainLayout = global.getMainLayout();

        menuContactTable = new MenuContactTable();
        menuFolderTable = new MenuFolderTable();
        menuAdmin = new MenuAdmin();

        CAPTION_TAB_FILE_MANAGER = Global.i("CAPTION_FILE_MANAGER");
        CAPTION_TAB_CONTACTS = Global.i("CAPTION_CONTACTS");
        CAPTION_TAB_ADMIN = Global.i("CAPTION_ADMIN");

        setSpacing(true);

        // Panel qui contiendra les dossiers
        PanelLight pnlFileManager = new PanelLight();
        // Contenu du panel
        final SQLTable table = new SQLTable();
        table.setStyleName("no-resizable");
        table.setSizeFull();
        table.setColumnHeaderMode(Table.COLUMN_HEADER_MODE_HIDDEN);
        table.addContainerProperty("bt_label", Label.class, null,
                Global.i("CAPTION_FOLDER"), null, null);
        table.addListener(new ItemClickListener() {
```



```java
            public void itemClick(ItemClickEvent event) {
                if (event.getButton() == ItemClickEvent.BUTTON_LEFT) {
                    String[] ids = table.getValuesFromObject(table.getValue());
                    if (ids.length <= 1) {
                        table.setValue(null);
                    } // Dé-sélectionne tout
                    table.select(event.getItemId()); // Sélectionne la ligne en
                                                     // cours
                    getWindow().showNotification(table.getValue().toString());
                }
            }
        });

        pnlFileManager.addComponent(menuFolderTable);
        global.setMenuFolderTable(menuFolderTable);

        // Panel
        PanelLight pnlContacts = new PanelLight();
        pnlContacts.addComponent(menuContactTable);
        global.setMenuContactTable(menuContactTable);

        // Panel admin
        PanelLight pnlAdmin = new PanelLight();
        pnlAdmin.addComponent(menuAdmin);
        global.setMenuAdmin(menuAdmin);

        // Menu Accordion
        accMenu = new Accordion();
        accMenu.setStyleName("v-accordion-icon");
        accMenu.setSizeFull();
        // Tab 1 (FileManager)
        ThemeResource iconManager = new ThemeResource(
                Global.PATH_THEME_RESSOURCES + "clouddrive.png");
        accMenu.addTab(pnlFileManager, CAPTION_TAB_FILE_MANAGER, iconManager);
        // Tab 2 (contacts)
        ThemeResource iconContact = new ThemeResource(
                Global.PATH_THEME_RESSOURCES + "contact.png");
        accMenu.addTab(pnlContacts, CAPTION_TAB_CONTACTS, iconContact);
        // Tab 3 (admin)
        if (UserSession.isAdmin()) {
            ThemeResource iconAdmin = new ThemeResource(
                    Global.PATH_THEME_RESSOURCES + "admin.png");
            accMenu.addTab(pnlAdmin, CAPTION_TAB_ADMIN, iconAdmin);
        }

        accMenu.addListener(this);

        // S'il n'existe encore aucun contact pour cet utilisateur
        if (tbl_contacts.getNumberOfContact() < 1) {
            accMenu.setSelectedTab(pnlContacts);
            selectedTab = CAPTION_TAB_CONTACTS;
        }

        addComponent(accMenu);
    }

    @Override
    public void selectedTabChange(SelectedTabChangeEvent event) {
        TabSheet tabsheet = event.getTabSheet();
        Tab tab = tabsheet.getTab(tabsheet.getSelectedTab());
        if (tab != null) {
            selectTab(tab.getCaption());
```

```
150         }
151     }
152
153     /**
154      * Sélectionne un onglet du menu
155      *
156      * @param caption
157      *            Le texte de l'onglet à sélectionner
158      */
159     public static void selectTab(String caption) {
160         Module slctModule = null;
161         if (caption.equals(CAPTION_TAB_FILE_MANAGER)) {
162             slctModule = new Module(TransfertModule.getBodyRibbonContent(),
163                     TransfertModule.getBodyContent());
164             global.getMenuFolderTable().loadContentForSelectedItem();
165             selectedTab = CAPTION_TAB_FILE_MANAGER;
166         } else if (caption.equals(CAPTION_TAB_CONTACTS)) {
167             if (tbl_contacts.getNumberOfContact() < 1) {
168                 slctModule = new Module(ContactModule.getBodyRibbonContent(),
169                         TransfertModule.getBodyContent());
170             } else {
171                 slctModule = new Module(ContactModule.getBodyRibbonContent(),
172                         ContactModule.getBodyContent());
173                 menuContactTable.reload();
174             }
175             selectedTab = CAPTION_TAB_CONTACTS;
176         } else if (caption.equals(CAPTION_TAB_ADMIN)) {
177             slctModule = new Module(AdminUsersModule.getBodyRibbonContent(),
178                     AdminUsersModule.getBodyContent());
179             selectedTab = CAPTION_TAB_ADMIN;
180         } else {
181             slctModule = new Module(TransfertModule.getBodyRibbonContent(),
182                     TransfertModule.getBodyContent());
183         }
184         CccvsTransfert.loadModule(slctModule);
185     }
186
187     /**
188      * Retourne l'accordéon (composant du menu)
189      *
190      * @return Menu latéral
191      */
192     public static Accordion getMenu() {
193         return accMenu;
194     }
195
196     /**
197      * Retourne le caption de l'onglet sélectionné dans le menu
198      *
199      * @return Caption de l'onglet sélectionné du menu
200      */
201     public static String getSelectedTab() {
202         return selectedTab;
203     }
204
205 }
206
```



```
1 package common.component;
2
3 import global.Global;
43
44 /**
45  * Table pour la gestion des contacts. Cette class permet l'affichage des
46  * contacts dans un tableau. Le tableau implémente un menu, affiché sur le clic
47  * droit de la souris. Ce menu propose la suppression et l'édition des contact.
48  * En sélection multiple dans le tableau, plusieurs contacts peuvent être
49  * supprimés en même temps, mais seule la ligne sur laquelle pointe le curseur
50  * est éditée lorsqu'il s'agit d'une édition. Lorsque l'édition est choisie, les
51  * informations sont transmises vers la class ContactForm.
52  *
53  * @author Dominique Roduit
54  * @version 1.0
55  *
56  */
57 public class ContactTable extends SQLTable {
58     /**
59      * Contient toutes les instances qui doivent être accessibles dans toute
60      * l'application
61      **/
62     protected GlobalObjects global = CccvsTransfert.getGlobalMethod();
63     /** Instance du bouton d'ajout d'un contact **/
64     private Button btAddContact = global.getBtAddContact();
65     /** Instance de la liste des contacts du menu **/
66     private MenuContactTable menuContactTable = global.getMenuContactTable();
67     /** Action éditer du menu contextuel **/
68     protected static Action ACTION_EDIT;
69     /** Action supprimer du menu contextuel **/
70     protected static Action ACTION_DEL;
71     /**
72      * Liste des actions du menu contextuel affiché sur le clique droit pour ce
73      * tableau
74      **/
75     protected static Action[] ACTIONS;
76     /** Contient la/les lignes du tableau sélectionnées **/
77     protected Object selectedItems = getValue();
78
79     /**
80      * Affiche un tableau contenant la liste des contacts
81      */
82     public ContactTable() {
83         super();
84
85         ACTION_EDIT = new Action(Global.i("CAPTION_EDIT"), new ThemeResource(
86                 Global.PATH_THEME_RESSOURCES + "action-edit-contact.png"));
87         ACTION_DEL = new Action(Global.i("CAPTION_DELETE"), new ThemeResource(
88                 Global.PATH_THEME_RESSOURCES + "false.png"));
89         ACTIONS = new Action[] { ACTION_EDIT, ACTION_DEL };
90
91         setSelectable(true);
92         setImmediate(false);
93         setMultiSelect(true);
94         setStyleName(Runo.TABLE_SMALL);
95
96         setSizeFull();
97         addContainerProperty("contact_icon", Embedded.class, null, "", null,
98                 Table.ALIGN_CENTER);
99         addContainerProperty("contact_name", String.class, null,
100                Global.i("CAPTION_NAME"), null, null);
101        addContainerProperty("contact_email", String.class, null,
```

```
102                 Global.i("CAPTION_ADRESSE_EMAIL"), null, null);
103         addContainerProperty("contact_type", String.class, null,
104                 Global.i("CAPTION_TYPE"), null, Table.ALIGN_CENTER);
105         addContainerProperty("contact_date", String.class, null,
106                 Global.i("CAPTION_LAST_MODIFICATION"), null, Table.ALIGN_CENTER);
107
108         setColumnWidth("contact_icon", 23);
109         setColumnWidth("contact_type", 70);
110         setColumnWidth("contact_date", 190);
111
112         // Gestion du clique droit
113         addActionHandler(new Action.Handler() {
114             // Sur le clique droit, on retourne les actions désirées
115             public Action[] getActions(Object target, Object sender) {
116                 if (target != null) {
117                     return ACTIONS;
118                 } else {
119                     return null;
120                 }
121             }
122
123             // Lors de l'appui sur une action
124             public void handleAction(Action action, Object sender, Object target) {
125                 if (action == ACTION_DEL) {
126                     actionDelete(target);
127                 } else if (action == ACTION_EDIT) {
128                     actionEdit(target);
129                 }
130             }
131         });
132
133         // Pour sélectionner la ligne par clique droit
134         addListener(new ItemClickListener() {
135             public void itemClick(ItemClickEvent event) {
136                 if (event.getButton() == ItemClickEvent.BUTTON_RIGHT) {
137                     String[] ids = getValuesFromObject(getValue());
138                     if (ids.length <= 1) {
139                         setValue(null);
140                     } // Dé-sélectionne tout
141                     select(event.getItemId()); // Sélectionne la ligne en cours
142
143                     selectedItems = getValue();
144                 }
145             }
146         });
147
148         reload();
149     }
150
151     /**
152      * Sélectionn des données qui vont remplir notre tableau
153      *
154      * @return (Object[]) Tableau d'objet contenant les lignes du tableau
155      */
156     public Object[] getTableData() {
157         super.PKField.clear();
158         ResultSet data = Sql.query(tbl_contacts.getSelectAllContactQuery());
159
160         Object[] files = null;
161         try {
162             data.last();
163             files = new Object[data.getRow() + 1];
```



```
164             data.beforeFirst();
165             int j = 0;
166
167             while (data.next()) {
168                 super.PKField.add(data.getInt("PKNoContact"));
169
170                 String interne_caption = (data.getBoolean("contact_internal")) ?
    Global
171                         .i("CAPTION_INTERNAL") : Global.i("CAPTION_EXTERNAL");
172                 String imgSrc = (data.getBoolean("contact_internal")) ? "home.png"
173                         : "icon-contact.png";
174                 Embedded imgContact = new Embedded(null, new ThemeResource(
175                         Global.PATH_THEME_RESSOURCES + imgSrc));
176                 files[j] = new Object[] {
177                         imgContact,
178                         data.getString("contact_name"),
179                         data.getString("contact_mail"),
180                         interne_caption,
181                         Utilities.formatSQLDate(data
182                                 .getString("contact_creation_date")) };
183                 j++;
184             }
185         } catch (SQLException e) {
186             e.printStackTrace();
187         } finally {
188             Sql.Disconnect();
189         }
190         return files;
191     }
192
193     /**
194      * (Re)Chargement des données du tableau
195      */
196     public void reload() {
197         super.reload(getTableData());
198     }
199
200     /**
201      * Action exécutée sur le clique de l'option "Supprimer" du menu contextuel
202      *
203      * @param target
204      *            (Object) Item en cours
205      */
206     public void actionDelete(Object target) {
207         String[] ids = getValuesFromObject(selectedItems);
208         for (int i = 0; i < ids.length; i++) {
209             Sql.exec(tbl_contacts.getDeleteContactQuery(ids[i]));
210         }
211         Sql.Disconnect();
212         reload();
213         reloadChildren();
214     }
215
216     /**
217      * Action exécutée sur le clique de l'option "Editer" du menu contextuel
218      *
219      * @param target
220      *            (Object) Item en cours
221      */
222     public void actionEdit(Object target) {
223         String[] ids = getValuesFromObject(selectedItems);
224         if (ids.length > 1) {
```

```
            setValue(null); // Déselectionne tout
            setValue(target); // Sélectionne uniquement la ligne en cours
            select(target); // Affiche le sélecteur sur la ligne en cours
            selectedItems = getValue();
        }

        // Clique automatique du bouton Ajouter un contact
        btAddContact.setStyleName(selectedItems.toString());
        btAddContact.click();
    }

    /**
     * Répercute le rechargement du contenu sur les enfants de cette classe
     */
    private void reloadChildren() {
        // On ajoute le lien avec la table contact du menu
        menuContactTable = global.getMenuContactTable();
        if (menuContactTable != null) {
            menuContactTable.reload();
        }
    }
}

```



```
package common.component;

import java.util.ArrayList;

/**
 * Menu affiché uniquement aux administrateurs de l'application.<br>
 * Il permet la gestion des lis
 *
 * @author Dominique Roduit
 *
 */
public class MenuAdmin extends CustomComponent {
    /** Layout principal qui englobe tout les composants **/
    private VerticalLayout mainLayout = new VerticalLayout();
    /** Liste des boutons du menu **/
    private ArrayList<Button> btMenu;
    /** Module chargé sur le clic des boutons **/
    private Module slctModule;

    /**
     * Création des composants du menu d'administration
     */
    public MenuAdmin() {
        mainLayout.setSizeFull();
        mainLayout.setSpacing(true);
        mainLayout.setMargin(true);
        mainLayout.setStyleName("admin-menu");

        // Liste des boutons du menu
        btMenu = new ArrayList<Button>();

        btMenu.add(new Button(Global.i("CAPTION_USERS"),
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        slctModule = new Module(AdminUsersModule
                                .getBodyRibbonContent(), AdminUsersModule
                                .getBodyContent());
                    }
                }));
        btMenu.get(0).setStyleName(Runo.BUTTON_DEFAULT);
        btMenu.get(0).setIcon(
                new ThemeResource(Global.PATH_THEME_RESSOURCES + "users.png"));

        btMenu.add(new Button(Global.i("CAPTION_JOURNALISATIONS"),
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        slctModule = new Module(AdminJournModule
                                .getBodyRibbonContent(), AdminJournModule
                                .getBodyContent());
                    }
                }));
        btMenu.get(1).setIcon(
                new ThemeResource(Global.PATH_THEME_RESSOURCES + "chart.png"));

        // Insertion des boutons
        for (Button button : btMenu) {
            button.setWidth("100%");
            button.addListener(new Button.ClickListener() {
                public void buttonClick(ClickEvent event) {
                    deselectAllButtons();
                    event.getButton().setStyleName(Runo.BUTTON_DEFAULT);

```

```
                    if (slctModule != null) {
                        CccvsTransfert.loadModule(slctModule);

                        if (event.getButton().getCaption()
                                .equals(Global.i("CAPTION_JOURNALISATIONS"))) {
                            GlobalObjects global = CccvsTransfert
                                    .getGlobalMethod();
                            global.getMainLayout().getBodyRibbonWrapper()
                                    .setHeight("90px");
                        }
                    }
                }
            });
            mainLayout.addComponent(button);
            mainLayout.setComponentAlignment(button, Alignment.MIDDLE_CENTER);
        }

        setCompositionRoot(mainLayout);
    }

    /**
     * Déselectionne tous les boutons du menu (par changement de style)
     */
    private void deselectAllButtons() {
        for (Button button : btMenu) {
            button.removeStyleName(Runo.BUTTON_DEFAULT);
        }
    }
}
```



```
package common.component;

import global.Global;

/**
 * Table pour la gestion des contacts, affichée dans le menu latéral gauche.
 * Cette class permet l'affichage des contacts dans un tableau. Le tableau
 * implémente un menu, affiché sur le clic droit de la souris. Ce menu propose
 * la suppression des contacts. En sélection multiple dans le tableau, plusieurs
 * contacts peuvent être supprimés en même temps.
 *
 * @author Dominique Roduit
 * @version 1.0
 *
 */
public class MenuContactTable extends ContactTable {
    /** Action supprimer du menu contextuel **/
    protected static Action ACTION_DEL;
    /**
     * Liste des actions du menu contextuel affiché sur le clique droit pour ce
     * tableau
     **/
    protected static Action[] ACTIONS;
    /** Stockage de la table des contacts du corps de page **/
    protected ContactTable contactTable = global.getTableContact();

    /**
     * Affiche un tableau contenant la liste des contacts
     */
    public MenuContactTable() {
        super();

        ACTION_DEL = new Action(Global.i("CAPTION_DELETE"), new ThemeResource(
                Global.PATH_THEME_RESSOURCES + "false.png"));
        ACTIONS = new Action[] { ACTION_DEL };

        setStyleName("no-resizable");

        removeContainerProperty("contact_name");
        addContainerProperty("contact_name", Label.class, null,
                Global.i("CAPTION_NAME"), null, null);
        setVisibleColumns(new Object[] { "contact_name" });

        removeAllActionHandlers();
        removeAllItems();

        // Gestion du clique droit
        addActionHandler(new Action.Handler() {
            // Sur le clique droit, on retourne les actions désirées
            public Action[] getActions(Object target, Object sender) {
                if (target != null) {
                    return ACTIONS;
                } else {
                    return null;
                }
            }

            // Lors de l'appui sur une action
            public void handleAction(Action action, Object sender, Object target) {
                if (action == ACTION_DEL) {
                    actionDelete(target);
                }
```

```
            }
        });

        // Sur le clique d'un contact
        addListener(new ItemClickListener() {
            public void itemClick(ItemClickEvent event) {
                if (event.getButton() == ItemClickEvent.BUTTON_LEFT) {
                    // getWindow().showNotification(event.getItemId().toString());
                }
            }
        });
    }

    /**
     * Sélectionn des données qui vont remplir notre tableau
     *
     * @return (Object[]) Tableau d'objet contenant les lignes du tableau
     */
    public Object[] getTableData() {
        super.PKField.clear();
        ResultSet data = Sql.query(tbl_contacts.getSelectAllContactQuery());

        System.out.println(Sql.rowCount() + " --- "
                + tbl_contacts.getSelectAllContactQuery());

        Object[] files = null;
        try {
            data.last();
            files = new Object[data.getRow() + 1];
            data.beforeFirst();
            int j = 0;

            while (data.next()) {
                super.PKField.add(data.getInt("PKNoContact"));

                Label lblContact = new Label(data.getString("contact_name"));
                lblContact.setStyleName("cursor-pointer");
                lblContact.addStyleName("v-menu-item-contact");
                if (!data.getBoolean("contact_internal")) {
                    lblContact.addStyleName("contact-extern");
                }

                files[j] = new Object[] { lblContact };
                j++;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            Sql.Disconnect();
        }
        return files;
    }

    /**
     * Action exécutée sur le clique de l'option "Supprimer" du menu contextuel
     *
     * @param target
     *            (Object) Item en cours
     */
    @Override
    public void actionDelete(Object target) {
        super.actionDelete(target);
```



```
        contactTable = global.getTableContact();
        contactTable.reload();
    }
}
```

```java
package common.component;

import global.Global;

/**
 * Table pour la gestion des dossiers. Cette class permet l'affichage des
 * dossiers dans un tableau. Le tableau implémente un menu, affiché sur le clic
 * droit de la souris. Ce menu propose la suppression des dossiers expirés et le
 * téléchargement des dossiers toujours disponibles.
 *
 * @author Dominique Roduit
 * @version 1.0
 *
 */
public class MenuFolderTable extends SQLTable {
    /**
     * Contient toutes les instances qui doivent être accessibles dans toute
     * l'application
     **/
    protected GlobalObjects global = CccvsTransfert.getGlobalMethod();
    /** Instance du bouton d'ajout d'un contact **/
    private Button btAddContact = global.getBtAddContact();
    /** Instance de la liste des contacts du menu **/
    private MenuContactTable menuContactTable = global.getMenuContactTable();
    /** Action supprimer du menu contextuel **/
    protected static Action ACTION_DEL;
    /** Action téléchargement **/
    protected static Action ACTION_DOWNLOAD;
    /** Action pour le renvoi d'un lien **/
    protected static Action ACTION_REMAIL;
    /** Action pour la récupération du lien de téléchargement **/
    protected static Action ACTION_GET_LINK;
    /**
     * Liste des actions du menu contextuel affiché sur le clique droit pour ce
     * tableau
     **/
    protected static Action[] ACTIONS;
    /** Contient la/les lignes du tableau sélectionnées **/
    protected Object selectedItems = getValue();
    /** Nom du dossier sélectionné **/
    private String folder_name = "";
    /** Date de création du dossier **/
    private String folder_creation_date = "";

    /**
     * Affiche un tableau contenant la liste des contacts
     */
    public MenuFolderTable() {
        super();

        ACTION_DEL = new Action(Global.i("CAPTION_DELETE"), new ThemeResource(
                Global.PATH_THEME_RESSOURCES + "false.png"));
        ACTION_DOWNLOAD = new Action(Global.i("CAPTION_DOWNLOAD"),
                new ThemeResource(Global.PATH_THEME_RESSOURCES + "down.png"));
        ACTION_REMAIL = new Action(Global.i("CAPTION_SEND_BACK_MAIL") + "...",
                new ThemeResource(Global.PATH_THEME_RESSOURCES
                        + "action-mail-send.png"));
        ACTION_GET_LINK = new Action(Global.i("CAPTION_GET_DOWNLOAD_LINK"),
                new ThemeResource(Global.PATH_THEME_RESSOURCES + "link.png"));
        ACTIONS = new Action[] { ACTION_DEL };

        setSelectable(true);
```



```java
        setImmediate(false);
        setMultiSelect(true);

        setStyleName("no-resizable");

        setSizeFull();
        addContainerProperty("folder_name", Label.class, null,
                Global.i("CAPTION_FOLDER"), null, null);

        // Gestion du clique droit
        addActionHandler(new Action.Handler() {
            // Sur le clique droit, on retourne les actions désirées
            public Action[] getActions(Object target, Object sender) {
                if (target != null) {
                    int item = Integer.parseInt(target.toString());
                    boolean archive = false;
                    ResultSet data = Sql.query(tbl_folders
                            .getSelectFolderInfos(item));
                    try {
                        if (data.first()) {
                            archive = data.getBoolean("folder_archive");
                        }
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }

                    if (archive) {
                        return ACTIONS;
                    } else {
                        return new Action[] { ACTION_DOWNLOAD, ACTION_GET_LINK,
                                ACTION_REMAIL };
                    }
                } else {
                    return null;
                }
            }

            // Lors de l'appui sur une action
            public void handleAction(Action action, Object sender, Object target) {
                int item = Integer.parseInt(target.toString());

                // Action "Supprimer"
                if (action == ACTION_DEL) {
                    actionDelete(target);
                }
                // Action "Télécharger"
                if (action == ACTION_DOWNLOAD) {
                    ResultSet data = Sql.query(tbl_folders
                            .getSelectFolderInfos(item));
                    try {
                        if (data.first()) {
                            folder_name = data.getString("folder_name");
                            folder_creation_date = data
                                    .getString("folder_creation_date");
                        }
                    } catch (SQLException e) {
                        e.printStackTrace();
                    }

                    String archiveName = Utilities.getFolderArchiveName(
                            folder_name, folder_creation_date);
```

```
177                     System.out.println(archiveName);
178
179                     File internFile = new File(Global.UPLOAD_DIR + archiveName);
180                     if (internFile.exists()) {
181                         CccvsTransfert.mainWindow.open(new ExternalResource(
182                                 Utilities.getURLForFile(archiveName)), "_self");
183                     } else {
184                         CccvsTransfert.mainWindow.showNotification(
185                                 Global.i("CAPTION_INEXISTING_FILE"),
186                                 Notification.TYPE_WARNING_MESSAGE);
187                     }
188
189                 }
190                 // Action *Renvoyer le lien aux destinataires*
191                 if (action == ACTION_REMAIL) {
192                     Window winRemail = new Window(
193                             Global.i("TITLE_WINDOW_REMAILLINK"));
194                     winRemail.setIcon(new ThemeResource(
195                             Global.PATH_THEME_RESSOURCES
196                                     + "action-mail-send.png"));
197                     winRemail.setResizable(false);
198                     winRemail.setModal(true);
199                     winRemail.addComponent(new ReMailLinkModule(item));
200                     winRemail.setWidth("700px");
201                     winRemail.setHeight("450px");
202                     global.setWinReMailLink(winRemail);
203                     CccvsTransfert.mainWindow.addWindow(winRemail);
204                 }
205                 // Action "Récupérer le lien"
206                 if (action == ACTION_GET_LINK) {
207                     ResultSet data = Sql.query(tbl_folders
208                             .getSelectFolderInfos(item));
209                     try {
210                         if (data.first()) {
211                             folder_name = data.getString("folder_name");
212                             folder_creation_date = data
213                                     .getString("folder_creation_date");
214                         }
215                     } catch (SQLException e) {
216                         e.printStackTrace();
217                     }
218
219                     String archiveName = Utilities.getFolderArchiveName(
220                             folder_name, folder_creation_date);
221
222                     global.openWindowGetLink(archiveName);
223                 }
224             }
225         });
226
227         // Pour sélectionner la ligne par clique droit
228         addListener(new ItemClickListener() {
229             public void itemClick(ItemClickEvent event) {
230                 if (event.getButton() == ItemClickEvent.BUTTON_RIGHT) {
231                     String[] ids = getValuesFromObject(getValue());
232                     if (ids.length <= 1) {
233                         setValue(null);
234                     } // Dé-sélectionne tout
235                     select(event.getItemId()); // Sélectionne la ligne en cours
236                 }
237
238                 selectedItems = getValue();
```



```
239                 CccvsTransfert.loadModule(new Module(TransfertModule
240                         .getBodyRibbonContent(), FolderModule
241                         .getBodyContent(Integer.parseInt(event.getItemId()
242                                 .toString()))));
243             }
244         });
245
246         reload();
247     }
248
249     /**
250      * Sélectionn des données qui vont remplir notre tableau
251      *
252      * @return (Object[]) Tableau d'objet contenant les lignes du tableau
253      */
254     public Object[] getTableData() {
255         super.PKField.clear();
256         ResultSet data = Sql.query(tbl_folders.getAllFolders());
257
258         Object[] files = null;
259         try {
260             data.last();
261             files = new Object[data.getRow() + 1];
262             data.beforeFirst();
263             int j = 0;
264
265             while (data.next()) {
266                 super.PKField.add(data.getInt("PKNoFolder"));
267
268                 Label lblDossier = new Label(data.getString("folder_name"));
269                 lblDossier.setStyleName("cursor-pointer");
270                 lblDossier.addStyleName("v-menu-item-folder");
271                 if (data.getBoolean("folder_archive")) {
272                     lblDossier.addStyleName("archive");
273                 }
274
275                 files[j] = new Object[] { lblDossier };
276                 j++;
277             }
278         } catch (SQLException e) {
279             e.printStackTrace();
280         } finally {
281             Sql.Disconnect();
282         }
283         return files;
284     }
285
286     /**
287      * (Re)Chargement des données du tableau
288      */
289     public void reload() {
290         super.reload(getTableData());
291     }
292
293     /**
294      * Action exécutée sur le clique de l'option "Supprimer" du menu contextuel
295      *
296      * @param target
297      *            (Object) Item en cours
298      */
299     public void actionDelete(Object target) {
300         String[] ids = getValuesFromObject(selectedItems);
```

```
301         for (int i = 0; i < ids.length; i++) {
302             Sql.exec(tbl_folders.getDeleteFolder(ids[i]));
303         }
304         Sql.Disconnect();
305         reload();
306         reloadChildren();
307 
308         // S'il n'y a aucun item dans la table
309         CccvsTransfert.loadModule(new Module(null, TransfertModule
310                 .getBodyContent()));
311     }
312 
313     /**
314      * Action exécutée sur le clique de l'option "Editer" du menu contextuel
315      *
316      * @param target
317      *            (Object) Item en cours
318      */
319     public void actionEdit(Object target) {
320         String[] ids = getValuesFromObject(selectedItems);
321         if (ids.length > 1) {
322             setValue(null); // Déselectionne tout
323             setValue(target); // Sélectionne uniquement la ligne en cours
324             select(target); // Affiche le sélecteur sur la ligne en cours
325             selectedItems = getValue();
326         }
327 
328         // Clique automatique du bouton Ajouter un contact
329         btAddContact.setStyleName(selectedItems.toString());
330         btAddContact.click();
331     }
332 
333     /**
334      * Répercute le rechargement du contenu sur les enfants de cette classe
335      */
336     private void reloadChildren() {
337         // On ajoute le lien avec la table contact du menu
338         menuContactTable = global.getMenuContactTable();
339         if (menuContactTable != null) {
340             menuContactTable.reload();
341         }
342     }
343 
344     /**
345      * Charge le contenu pour l'item sélectionné
346      */
347     public void loadContentForSelectedItem() {
348         String[] ids = getValuesFromObject(getValue());
349         if (!ids[0].equals("")) {
350             CccvsTransfert.loadModule(new Module(TransfertModule
351                     .getBodyRibbonContent(), FolderModule
352                     .getBodyContent(Integer.parseInt(ids[0]))));
353         }
354     }
355 }
356 
```



```
1 package common.component;
2 
3 import com.vaadin.ui.Panel;
6 
7 /**
8  * Création d'un panel dépourvu du style CSS natif de Vaadin
9  *
10  * @author Dominique Roduit
11  *
12  */
13 public class PanelLight extends Panel {
14     /** Layout pour la suppression des styles appliqués sur le Panel **/
15     private VerticalLayout layout;
16 
17     /**
18      * Création d'un Panel sans style CSS
19      */
20     public PanelLight() {
21         super();
22         setStyleName(Reindeer.PANEL_LIGHT);
23         setSizeFull();
24 
25         layout = (VerticalLayout) getContent();
26         layout.setStyleName("v-menu-buttons");
27         layout.setSizeFull();
28         layout.setMargin(false);
29         layout.setSpacing(false);
30     }
31 
32     /**
33      * Spécifications des marges
34      *
35      * @param enabled
36      *            true = Marges activées
37      */
38     public void setMargin(boolean enabled) {
39         layout.setMargin(enabled);
40     }
41 
42     /**
43      * Spécification des marges intérieur (padding)
44      *
45      * @param enabled
46      *            true = marges intérieur activées (padding)
47      */
48     public void setSpacing(boolean enabled) {
49         layout.setSpacing(enabled);
50     }
51 }
52 
```

```java
package common.component;

import java.util.ArrayList;

/**
 * Extension du composant Table de Vaadin.<br>
 * Cette table contient des méthodes utiles pour le chargement de données depuis
 * une base de données,<br>
 * il faut donc l'utiliser lorsqu'on souhaite afficher les données d'une table
 * de la base de données dans un tableau de l'interface utilisateur. Cette class
 * est une alternative à l'add-on Vaadin "SQLContainer" qui est complexe et
 * propose beaucoup de méthodes dont je n'ai pas l'utilité.<br>
 * Contrairement au SQLContainer, les requêtes SQL exécutées pour le chargement
 * de ce type de table contiennent directement les clauses ORDER BY, LIMIT,
 * etc...
 *
 * @author Dominique Roduit
 *
 */
public class SQLTable extends Table {
    protected ArrayList<Integer> PKField = new ArrayList<Integer>();

    /**
     * Création d'une table destinée à l'affichage d'un résultat d'une requête
     * SQL
     */
    public SQLTable() {
        super();
        setSizeFull();
        setSelectable(true);
        setImmediate(false);
        setMultiSelect(true);
    }

    /**
     * Ajoute tout le contenu d'un tableau d'objet (Items) dans le container
     *
     * @param items
     *            (Object[]) Résultats de la requête SQL sous forme d'items
     *            prêts à l'utilisation.
     */
    public void addAllItems(Object[] items) {
        if (items != null) {
            Item item = null;
            for (int i = 0; i < items.length; i++) {
                if (items[i] != null) {
                    addItem((Object[]) items[i], PKField.get(i));
                    setData(PKField.get(i));
                }
            }
        }
    }

    /**
     * Mise à jour des données du tableau
     *
     * @param items
     *            Lignes de la table
     */
    public void reload(Object[] items) {
        if (removeAllItems()) {
            addAllItems(items);
```



```java
        }
        // S'il n'y a encore aucune ligne on affiche pas le tableau
        if (size() < 1) {
            setVisible(false);
        } else {
            setVisible(true);
        }
    }

    /**
     * Converti un objet en tableau de String.<br>
     * Utilisé pour convertir l'objet "value" en tableau de chaine de caractères
     *
     * @param value
     *            Objet à passer en paramètre
     * @return (String[]) Tableau de string avec les valeurs
     */
    protected String[] getValuesFromObject(Object value) {
        String rowId = "";
        String[] ids = null;

        if (value != null) {
            rowId = value.toString();
            rowId = rowId.replace("[", "").replace("]", "").replaceAll(" ", "");
            ids = rowId.split(",");
        }

        return ids;
    }
}
```

PACKAGE

COMMON.COMPONENT.UPLOAD

- CustomUpload
- FileInfo
- FileUp
- MultipleFileUpload
- TempFileFactory2
- *UploadActionListener*

Composants, classes & interfaces utiles au transfert multiple de fichier.
Les classes contenus dans ce package sont des surcharges des classes du package original de l'add-on easyuploads.

```java
package common.component.upload;

import form.WizFoldUpload;

/**
 * Utilisation de la classe MultipleFileUpload, récupération et implémentation
 * des évènements de l'interface UploadActionListener. Gestion de l'affichage de
 * la progression et de la limitation de la taille des fichiers.
 *
 * @author Dominique Roduit
 *
 */
public class CustomUpload extends MultipleFileUpload {
    /**
     * Liste des fichiers à la fin de l'envoi, tels qu'ils ont été enregistrés
     * sur le disque
     **/
    private ArrayList<FileUp> listRenamedFiles = new ArrayList<FileUp>();
    /** Stockage des objets globals **/
    private GlobalObjects global = CccvsTransfert.getGlobalMethod();
    /** Fenêtre principale **/
    private Window mainWindow = CccvsTransfert.mainWindow;
    /** Tableau qui réceptionne les fichiers sélectionnés **/
    private Table uploadTable = global.getUploadTable();
    /** Indicateurs de progression **/
    private LinkedList<ProgressIndicator> indicators;
    /** Bouton "Terminer" **/
    private Button btFinish;
    /** Bouton "Annuler" **/
    private Button btCancel;
    /** Statut (0=en train d'envoyer, 1=libre) **/
    private int status = 0;
    /**
     * Détermine si le transfert est autorisé selon le type des fichiers et leur
     * tailles
     **/
    private boolean transfertAllowed = false;
    /** Module d'upload **/
    private CustomUpload mFileUp;

    /**
     * Gestion des évènements qui surviennent lorsque des fichiers sont
     * transférés
     */
    public CustomUpload(final Folder folder,
            final ArrayList<Integer> slctContacts) {
        super();

        setAllowedExtentions(Global.getParm("UPLOAD_ALLOWED_EXTENSIONS"));

        addUploadActionListener(new UploadActionListener() {

            @Override
            public void fileUploadStarted(int idFile, String filename,
                    int pendingFiles) {
                uploadTable.setValue(null);
                uploadTable.setValue(idFile);
                uploadTable.select(idFile);
                uploadTable.setCurrentPageFirstItemIndex(idFile);
            }

            @Override
```



```java
            public void fileUploadProgress(int idFile, String filename,
                    int pendingFiles, float progress) {

                String value = uploadTable.getValue().toString();
                value = value.replace("[", "").replace("]", "").trim();

                if (value.length() > 0 && value != null) {
                    Item item = uploadTable.getItem(Integer.parseInt(value));

                    String percents = "0%";
                    NumberFormat progression = NumberFormat.getNumberInstance();
                    progression.setMaximumFractionDigits(1);
                    percents = progression.format(progress).concat(" %");

                    item.getItemProperty("file_etat").setValue(percents);
                    indicators.get(Integer.parseInt(value) - 1).setValue(
                            progress / 100);
                }

            }

            @Override
            public void fileUploadFinished(int idFile, String filename,
                    File file, int pendingFiles) {
                String value = uploadTable.getValue().toString();
                value = value.replace("[", "").replace("]", "").trim();

                FileUp f = new FileUp(file, filename);
                f.setId(idFile);

                if (value.length() > 0 && value != null) {
                    Item item = uploadTable.getItem(Integer.parseInt(value));
                    item.getItemProperty("file_etat").setValue(
                            Global.i("CAPTION_FINISHED"));
                    indicators.get(Integer.parseInt(value) - 1).setValue(1);
                    indicators.get(Integer.parseInt(value) - 1).setStyleName(
                            "success");
                    listRenamedFiles.add(f);

                    float progress = Float.parseFloat(Integer
                            .toString(getAllowedFilesUploaded()))
                            / Float.parseFloat(Integer
                                    .toString(getInitListFile().size()));
                    WizFoldUpload.getSuperIndicator().setValue(progress);
                    if (progress >= 1) {
                        WizFoldUpload.getSuperIndicator().setVisible(false);
                    }
                }
            }

            @Override
            public void fileUploadError(String filename, int pendingFiles) {
                getWindow()
                        .showNotification("Erreur lors de l'envoi", filename);
            }

            @Override
            public void fileUploadComplete(ArrayList<File> fileList) {
                if (transfertAllowed) {
                    // On active le bouton Terminer
                    btFinish.setEnabled(true);
                    WizFoldUpload.getBtRename().setEnabled(true);
```

```
146                     WizFoldUpload.getSuperIndicator().setVisible(false);
147                 }
148                 // On rend les item de la table sélectionnables
149                 uploadTable.setSelectable(true);
150                 // Indique que l'upload est terminé
151                 status = 1;
152             }
153 
154             @Override
155             public void onSelectedFiles(int error) {
156                 System.out.println("Code erreur : " + error);
157                 System.out.println("Fichiers autorisés : "
158                         + getInitListFile().size());
159                 btFinish = WizFoldUpload.getBtFinish();
160                 btCancel = WizFoldUpload.getBtCancel();
161 
162                 transfertAllowed = ((error == 0 && getQueueSize() <= Integer
163                         .parseInt(Global.getParm("UPLOAD_MAX_SIZE"))) || (error ==
    2 && getInitListFile()
164                         .size() > 0)) ? true : false;
165 
166                 // On commence à uploader (0 = indisponible)
167                 if (transfertAllowed) {
168                     WizFoldUpload.getSuperIndicator().setVisible(true);
169                     WizFoldUpload
170                             .getLblSize()
171                             .setValue(
172                                     Utilities.formatSize(getQueueSize())
173                                             + " / "
174                                             +
    Utilities.formatSize(Long.parseLong(Global
175                                                     .getParm("UPLOAD_MAX_SIZE"))));
176                     status = 0;
177 
178                     // On cache le label indiquant de sélectionner les fichiers,
179                     // s'il existe
180                     if (WizFoldUpload.getSlctFileLbl() != null) {
181                         WizFoldUpload.getSlctFileLbl().setVisible(false);
182                     }
183 
184                     // On supprime le bouton Annuler
185                     btCancel.setEnabled(false);
186                     btCancel.setVisible(false);
187 
188                     // On affiche le bouton Terminer mais on ne l'active pas
189                     // encore
190                     btFinish.setVisible(true);
191                     btFinish.setEnabled(false);
192                     WizFoldUpload.getBtRename().setVisible(true);
193                     WizFoldUpload.getBtRename().setEnabled(false);
194 
195                     // On cache la zone HTML5 pour l'envoi par glissé-déposé
196                     setDropZoneVisible(false);
197 
198                     // On affiche la table qui affiche les upload en cours
199                     uploadTable.setVisible(true);
200                     uploadTable.setSelectable(false);
201 
202                     if (indicators == null) {
203                         indicators = new LinkedList<ProgressIndicator>();
204                     }
205 
```



```
206                     ArrayList<FileInfo> FileList = getInitListFile();
207                     for (FileInfo file : FileList) {
208                         System.out.println(file.getId() + " : "
209                                 + file.getName());
210 
211                         ProgressIndicator pi = createProgressIndicator();
212                         indicators.add(pi);
213 
214                         int indexNo = (indicators.get(file.getId() - 1) == null) ?
    file
215                                 .getId() : file.getId() - 1;
216 
217                         Object[] item = new Object[] {
218                                 // file.getId(),
219                                 Utilities.getImgFromExtension(Utilities
220                                         .getExtension(file.getName())),
221                                 file.getName(),
222                                 Utilities.formatSize(file.getSize()),
223                                 indicators.get(indexNo),
224                                 Global.i("CAPTION_QUEUED") };
225                         int itemId = file.getId();
226                         uploadTable.addItem(item, itemId);
227                     }
228                 }
229 
230                 // S'il y a une erreur
231                 switch (error) {
232                 case 1: // Si on a essayé d'envoyer la corbeille ou le bureau
233                         // (Uniquement par le drag&drop)
234                     mainWindow.showNotification(Global.i("CAPTION_ERROR")
235                             .toUpperCase(), Global.i("CAPTION_UPLOAD_ERR1"),
236                             Notification.TYPE_ERROR_MESSAGE);
237                     break;
238                 case 2: // S'il y a des fichiers dont l'extension n'est pas
239                         // autorisée
240                     ArrayList<String> filesNonAutorised = getNonAutorizedFiles();
241                     if (filesNonAutorized != null) {
242                         if (filesNonAutorized.size() > 0) {
243                             // Construction du message d'erreur
244                             String message = "";
245                             for (String file : filesNonAutorized) {
246                                 message += "- " + file + "<br>";
247                             }
248                             message += "<br><span style=\"color:#dff;\">"
249                                     + Global.i("CAPTION_EXTENSION_AUTHORISE")
250                                     + " : " + getAllowedExtentions().toString()
251                                     + "</span>";
252                             mainWindow.showNotification(
253                                     Global.i("CAPTION_FILE_NON_AUTHORIZED")
254                                             + " (" + filesNonAutorised.size()
255                                             + ")", message,
256                                     Notification.TYPE_TRAY_NOTIFICATION, true);
257                         }
258                     }
259                     break;
260                 case 3: // Si la taille totale des fichiers sélectionnés dépasse
261                         // 1 Go (Depuis le bouton Parcourir uniquement)
262                     mainWindow.showNotification(
263                             Global.i("CAPTION_LIMITED_SIZE"),
264                             Global.i("CAPTION_UPLOAD_ERR3"),
265                             Notification.TYPE_ERROR_MESSAGE);
266                     if (uploadTable.size() == 0) { // Si on a encore aucun
```

```
                                                // fichier dans le tableau
                        global.getWindowFolder().removeAllComponents();
                        global.getWindowFolder().addComponent(
                                new WizFoldUpload(folder, slctContacts));
                    } else {
                        mFileUp = global.getUploadModule();
                        global.getUploadModuleLayout().removeComponent(mFileUp);
                        uploadTable.setHeight("245px");
                    }
                    break;
                case 4: // Si la taille totale des fichiers sélectionnés dépasse
                        // 1 Go (DEPUIS LA DROPZONE !)
                    mainWindow.showNotification(
                            Global.i("CAPTION_LIMITED_SIZE"),
                            Global.i("CAPTION_UPLOAD_ERR3"),
                            Notification.TYPE_ERROR_MESSAGE);
                    break;
                }

                System.out.println(getQueueSize() + " Octets -- "
                        + Utilities.formatSize(getQueueSize()));

                // Si la taille des fichiers sélectionnés dépasse la taille
                // totale autorisée
                if ((getQueueSize() >= Integer.parseInt(Global
                        .getParm("UPLOAD_MAX_SIZE"))) && error == 0) {
                    mFileUp = global.getUploadModule();
                    global.getUploadModuleLayout().removeComponent(mFileUp);
                    WizFoldUpload.getBtRename().click();
                    WizFoldUpload.getLblSize().setValue(
                            "Taille maximum atteinte");
                }

            }
        });

    }

    /**
     * Fonction appelée lorsque l'upload d'un fichier est terminé (equivalent du
     * Listener "fileUploadFinished")
     *
     * @param file
     *            Informations sur le fichier uploadé
     * @param fileName
     *            Nom du fichier uploadé
     * @param mimeType
     *            Type MIME du fichier uploadé
     */
    @Override
    protected void handleFile(File file, String fileName, String mimeType,
            long length) {

    }

    /**
     * Création du buffer qui stock les données sur le disque
     */
    @Override
    protected FileBuffer createReceiver() {
        FileBuffer receiver = super.createReceiver();
        // S'assure que le récepteur ne supprime pas les fichiers après qu'ils
```



```
        // aient été manipulés par #handleFile()
        receiver.setDeleteFiles(false);
        return receiver;
    }

    /**
     * @return Obtention de la liste des fichiers tels qu'ils sont enregistrés à
     *         la fin de l'upload
     */
    public ArrayList<FileUp> getRenamedFileList() {
        return listRenamedFiles;
    }

    /**
     * @return Retourne le statut (0=Occupé, en envoi, 1=Libre, en attente de
     *         réception de fichiers)
     */
    public int getStatus() {
        return status;
    }

}

```

```java
package common.component.upload;

/**
 * Class intermédiaire dont le but est de stocker des informations précises sur
 * un fichier.<br>
 * On peut ensuite utiliser directement ce modèle pour récupérer tous les
 * attributs du fichier dont on a besoin.<br>
 * Utilisé exclusivement pour l'upload de fichier dans la class
 * {@link MultiFileUpload}.
 *
 * @author Dominique Roduit
 *
 */
public class FileInfo {
    /** Identifiant du fichier **/
    private int id = 0;
    /** Nom du fichier **/
    private String name = "";
    /** Taille du fichier **/
    private long size = 0;
    /** Type MIME du fichier **/
    private String type = "";

    /**
     * Création d'un model de fichier pour le stockage des informations d'un
     * fichier dans un objet
     *
     * @param id
     *            Identifiant du fichier
     * @param fileName
     *            Nom du fichier
     * @param contentLength
     *            Taille du fichier
     * @param mimeType
     *            Type MIME du fichier
     */
    public FileInfo(int id, String fileName, long contentLength, String mimeType) {
        this.id = id;
        name = fileName;
        size = contentLength;
        type = mimeType;
    }

    /**
     * @return Nom du fichier
     */
    public String getName() {
        return name;
    }

    /**
     * @return Taille du fichier
     */
    public long getSize() {
        return size;
    }

    /**
     * @return Type MIME du fichier
     */
    public String getType() {
        return type;
```



```java
    }

    /**
     * @return Identifiant du fichier
     */
    public int getId() {
        return id;
    }

    /**
     * @param id
     *            Identifiant du fichier
     */
    public void setId(int id) {
        this.id = id;
    }
}
```

```java
package common.component.upload;

import java.io.File;

/**
 * Model d'un fichier pour l'intégration dans le tableau d'upload. La class sert
 * à stocker les informations sur les fichiers sélectionnés dans le but de
 * pouvoir récupérer tous les attributs d'un fichier.
 *
 * @author Dominique Roduit
 *
 */
public class FileUp {
    /** Identifiant du fichier dans le tableau **/
    private int id;
    /** Informations sur le fichier enregistré sur le disque **/
    private File fileDiskInfo;
    /** Nom original du fichier (tel qu'il était nommé sur le PC de l'auteur) **/
    private String originalName;
    /** Nom du fichier renommé **/
    private String rename;
    /** Description du fichier **/
    private String description;

    /**
     * Création d'un model de fichier pour la table d'upload
     *
     * @param fileDiskInfo
     *            Informations sur le fichier enregistré sur le disque
     * @param originalName
     *            Nom original du fichier (tel qu'il était nommé sur le PC de
     *            l'auteur)
     */
    public FileUp(File fileDiskInfo, String originalName) {
        this.setFileDiskInfo(fileDiskInfo);
        this.setOriginalName(originalName);
    }

    /**
     * Création d'un model de fichier pour la table d'upload
     *
     * @param fileDiskInfo
     *            Informations sur le fichier enregistré sur le disque
     * @param originalName
     *            Nom original du fichier (tel qu'il était nommé sur le PC de
     *            l'auteur)
     * @param rename
     *            Nom du fichier renommé
     */
    public FileUp(File fileDiskInfo, String originalName, String rename) {
        this(fileDiskInfo, originalName);
        setRename(rename);
    }

    /**
     * Obtention du nom original du fichier
     *
     * @return Nom du fichier original
     */
    public String getOriginalName() {
        return originalName;
    }
```



```java

    /**
     * Enregistrement du nom original du fichier
     *
     * @param originalName
     */
    public void setOriginalName(String originalName) {
        this.originalName = originalName;
    }

    /**
     * Obtention des informations sur le fichier enregistré sur le disque
     *
     * @return Informations sur le fichier enregistré
     */
    public File getFileDiskInfo() {
        return fileDiskInfo;
    }

    /**
     * Enregistrement des informations du fichier enregistré
     *
     * @param fileDiskInfo
     *            informations du fichier enregistréI
     */
    public void setFileDiskInfo(File fileDiskInfo) {
        this.fileDiskInfo = fileDiskInfo;
    }

    /**
     * @return Description du fichier
     */
    public String getDescription() {
        return description;
    }

    /**
     * @param description
     *            Description du fichier
     */
    public void setDescription(String description) {
        this.description = description;
    }

    /**
     * @return Nom du fichier redéfinit par l'utilisateur
     */
    public String getRename() {
        return rename;
    }

    /**
     * @param Nom
     *            du fichier redéfinit par l'utilisateur
     */
    public void setRename(String rename) {
        this.rename = rename;
    }

    /**
     * @return Obtention de l'identifiant du fichier dans le tableau
     */
```

```
125     public int getId() {
126         return id;
127     }
128
129     /**
130      * @param id
131      *            ID du fichier dans le tableau
132      */
133     public void setId(int id) {
134         this.id = id;
135     }
136 }
137
```



```
1 package common.component.upload;
2
3 import global.Global;
52
53 /**
54  * MultipleFileUpload est la class principale qui permet le téléchargement de
55  * fichiers multiple. Elle est basée sur la class {@link MultiFileUpload} de
56  * Vaadin qui permet le téléchargement immédiat de plusieurs fichiers en
57  * parallèles. Elle affiche également les indicateurs de progression de l'envoi
58  * des fichiers ainsi qu'une zone de drag&drop pour les navigateurs qui
59  * acceptent cette technologie. <br>
60  * <br>
61  * Cette class enregistre les flux directement dans des fichiers pour limiter la
62  * consommation de mémoire. <br>
63  * Créé des fichiers temporaires par défaut, mais cela peut être modifié avec
64  * {@link #setFileFactory(FileFactory)} (par ex. pour cibler directement le
65  * répertoire serveur) TODO Temps restant estimé et taux de transfert
66  *
67  */
68 @SuppressWarnings("serial")
69 public abstract class MultipleFileUpload extends CssLayout implements
70         DropHandler {
71
72     private CssLayout progressBars = new CssLayout();
73     /** Contient les composants du plugin **/
74     private CssLayout uploads = new CssLayout();
75     /** Texte par défaut du bouton "Parcourir..." **/
76     private String uploadButtonCaption = "Parcourir...";
77     /** Texte de la zone drag&drop */
78     private String areaText = "<small>" + Global.i("CAPTION_DRAG_DROP")
79             + "</small>";
80     /** Liste contenant les fichiers envoyés et leurs infos */
81     private ArrayList<File> fileList = new ArrayList<File>();
82     /** Liste des fichiers juste après la sélection **/
83     private ArrayList<FileInfo> initFileList = new ArrayList<FileInfo>();
84     /** ID du fichier en cours d'envoi **/
85     private int currentFileId = 0;
86     /** ID du fichier envoyé **/
87     private int idFile = 0;
88     /** nombre d'upload effectués */
89     private long transfertNo = 0;
90     /** Taile total de tous les fichiers sélectionnés */
91     private long allFilesSize = 0;
92     /** Nombre de fichiers en attente à l'initialisation */
93     private int initPendingFilesNo = 0;
94     /** Nombre de fichiers en attente */
95     private int pendingFilesNo = 0;
96     /** Indique si l'upload d'un fichier est en cours */
97     private boolean isInProcess = false;
98     /** Nombre de fichiers autorisés qui ont étés envoyés **/
99     private int authorizedFilesUploadedNo = 0;
100
101     /** Taille maximal de la liste d'attente (tous les fichiers), défault : 1 Go */
102     private int maxQueueSize = Integer.parseInt(Global
103             .getParm("UPLOAD_MAX_SIZE"));
104
105     /** Contient la progression du fichier en cours */
106     private float progression = 0;
107     /** Contient la liste des extensions autorisées */
108     private ArrayList<String> allowedExtensions = new ArrayList<String>();
109     /** Stock le nom des fichiers non autorisés */
110     ArrayList<String> filesNonAutorized = new ArrayList<String>();
```

```
111
112     /** Tous les {@link UploadActionListener} enregistrés */
113     private List<UploadActionListener> uploadListeners = new
    Vector<UploadActionListener>();
114
115     public MultipleFileUpload() {
116         allowedExtensions.add("*");
117
118         addComponent(progressBars);
119         uploads.setStyleName("v-multifileupload-uploads");
120         addComponent(uploads);
121         prepareUpload();
122     }
123
124     /**
125      * Ajoute l'action spécifiée {@link UploadActionListener}. Tous les
126      * {@link UploadActionListener} enregistrés seront informés des actions de
127      * téléchargement des fichiers. Il faut supprimer le Listener à la fin de
128      * l'utilisation pour prévenir les fuites de mémoire
129      * <p />
130      * Si le Listener est déjà enregistré, rien ne se passera.
131      *
132      * @param l
133      *            Le Listener à ajouter
134      */
135     public void addUploadActionListener(UploadActionListener l) {
136         if (!uploadListeners.contains(l))
137             uploadListeners.add(l);
138     }
139
140     /**
141      * Supprime l'action {@link UploadActionListener} spécifiée. L'
142      * {@link UploadActionListener} ne sera plus informé sur les actions de
143      * téléchargement des fichiers If the listener is not registered nothing
144      * will be do.
145      *
146      * @param l
147      *            the listener to remove
148      */
149     public void removeUploadActionListener(UploadActionListener l) {
150         uploadListeners.remove(l);
151     }
152
153     /**
154      * Evenement déclenché lorsque les fichiers sont sélectionnés
155      *
156      * @param error
157      **/
158     private void notifyOnSelectedFiles(int error) {
159         for (UploadActionListener l : uploadListeners) {
160             l.onSelectedFiles(error);
161         }
162     }
163
164     /**
165      * Notifie tous les {@link UploadActionListener} du démarrage du processus
166      * d'envoi d'un fichier
167      **/
168     private void notifyUploadStart(int idFile, String filename, int pendingFiles) {
169         for (UploadActionListener l : uploadListeners) {
170             l.fileUploadStarted(idFile, filename, pendingFiles);
171
```



```
172             if (initPendingFilesNo - pendingFiles == 1) {
173                 System.out
174                         .println("\n#==============================================
    ==========#\n"
175                                 + "-------- Démarrage d'une session de transfère ("
176                                 + transfertNo
177                                 + ") -------\n"
178                                 + "-------- Taille total du transfert : "
179                                 + Utilities.formatSize(allFilesSize)
180                                 + "      -------\n"
181                                 + "-------- Extensions autorisées : "
182                                 + allowedExtensions.toString() + " -------\n");
183             }
184
185             System.out.println("#----------------- Upload ("
186                     + (initPendingFilesNo - pendingFiles) + "/"
187                     + initPendingFilesNo
188                     + ") ------------------#\n- Fichier : " + filename);
189         }
190     }
191
192     /**
193      * Notifie tous les {@link UploadActionListener} de la fin du processus
194      * d'envoi d'un fichier
195      **/
196     private void notifyUploadFinished(int idFile, String filename, File file,
197             int pendingFiles) {
198         for (UploadActionListener l : uploadListeners) {
199             l.fileUploadFinished(idFile, filename, file, pendingFiles);
200         }
201
202         System.out.println("- Taille : " + Utilities.formatSize(file.length())
203                 + "\n" + "- Fichiers restants : " + pendingFiles + "/"
204                 + initPendingFilesNo + "\n"
205                 + "#-------------------------------------------------#\n");
206         if (pendingFiles == 0) {
207             System.out
208                     .println("#==================================================
    ====#\n");
209         }
210     }
211
212     /** Notifie tous les {@link UploadActionListener} d'une erreur d'envoi */
213     private void notifyUploadError(String filename, int pendingFiles) {
214         for (UploadActionListener l : uploadListeners) {
215             l.fileUploadError(filename, pendingFiles);
216             System.out
217                     .println("#----------------- ERREUR ------------------#\n- "
218                             + filename
219                             + "\n- Fichiers restants : "
220                             + pendingFiles + "\n");
221         }
222     }
223
224     /**
225      * Evenement déclenché chaque fois que la progression de l'envoi d'un
226      * fichier augmente
227      **/
228     private void notifyUploadProgess(int idFile, String filename,
229             int pendingFiles, float progress) {
230         for (UploadActionListener l : uploadListeners) {
231             l.fileUploadProgress(idFile, filename, pendingFiles, progress);
```

```
232         }
233     }
234
235     /**
236      * Evenement déclenché lorsque le transfert de tous les fichiers de la file
237      * d'attente sont envoyés
238      **/
239     private void notifyUploadComplete(ArrayList<File> fileList) {
240         for (UploadActionListener l : uploadListeners) {
241             l.fileUploadComplete(fileList);
242         }
243         initPendingFilesNo = 0;
244     }
245
246     private MultiUpload upload;
247
248     /**
249      * Initialisation du plugin pour l'upload des fichiers (Sélection par bouton
250      * Parcourir...)
251      */
252     private void prepareUpload() {
253         final FileBuffer receiver = createReceiver();
254
255         upload = new MultiUpload();
256         MultiUploadHandler handler = new MultiUploadHandler() {
257             private LinkedList<ProgressIndicator> indicators;
258
259             /**
260              * Exécuté losrque l'envoi d'un fichier démarre
261              */
262             public void streamingStarted(StreamingStartEvent event) {
263                 isInProcess = true;
264                 pendingFilesNo--;
265
266                 if (!filesNonAutorized.contains(event.getFileName())) {
267                     currentFileId++;
268                     notifyUploadStart(currentFileId, event.getFileName(),
269                             pendingFilesNo);
270                 }
271             }
272
273             /**
274              * Exécuté lorsque l'envoi d'un fichier se termine
275              */
276             public void streamingFinished(StreamingEndEvent event) {
277                 File file = receiver.getFile();
278
279                 String originalFileName = file.getName();
280                 File newFile = null;
281                 // Renommage du fichier
282                 if (file.exists()) {
283                     file.setWritable(true);
284                     file.setReadable(true);
285                     file.setExecutable(true);
286
287                     String newName = Global.UPLOAD_DIR
288                             + Utilities.getNewFileName(originalFileName);
289                     // getWindow().showNotification("Enregistré sous", newName);
290                     newFile = new File(newName);
291
292                     file.renameTo(newFile);
293                 }
```



```
294                 fileList.add(newFile);
295
296                 handleFile(newFile, newFile.getName(), event.getMimeType(),
297                         event.getBytesReceived());
298                 receiver.setValue(null);
299
300                 isInProcess = false;
301
302                 // Suppression des fichiers non-autorisés
303                 if (filesNonAutorized.contains(originalFileName)) {
304                     newFile.delete();
305                 } else {
306                     // Incrémentation du nombre de fichier autorisé envoyé
307                     authorizedFilesUploadedNo++;
308                 }
309
310                 notifyUploadFinished(currentFileId, event.getFileName(),
311                         newFile, pendingFilesNo);
312
313                 if (authorizedFilesUploadedNo == (initPendingFilesNo -
    filesNonAutorized
314                         .size()) || pendingFilesNo == 0) {
315                     notifyUploadComplete(fileList);
316                     setEnabledUploadDropZone(true);
317                     setEnableUploadButton(true);
318                 }
319             }
320
321             /**
322              * Exécuté lorsque l'envoi d'un fichier échoue
323              */
324             public void streamingFailed(StreamingErrorEvent event) {
325                 Logger.getLogger(getClass().getName()).log(Level.FINE,
326                         "L'envoi à échoué", event.getException());
327
328                 isInProcess = false;
329
330                 notifyUploadError(event.getFileName(), pendingFilesNo);
331             }
332
333             /**
334              * Exécuté lors de la progression de l'envoi du fichier
335              */
336             public void onProgress(StreamingProgressEvent event) {
337                 long readBytes = event.getBytesReceived();
338                 long contentLength = event.getContentLength();
339                 float f = (float) readBytes / (float) contentLength;
340
341                 progression = (f * 100);
342                 notifyUploadProgess(currentFileId, event.getFileName(),
343                         pendingFilesNo, progression);
344
345                 isInProcess = true;
346             }
347
348             public OutputStream getOutputStream() {
349                 FileDetail next = upload.getPendingFileNames().iterator()
350                         .next();
351                 return receiver.receiveUpload(next.getFileName(),
352                         next.getMimeType());
353             }
354
```

```
355             /**
356              * Exécuté lorsque les fichiers à envoyer sont sélectionnés
357              */
358             public void filesQueued(Collection<FileDetail> pendingFileNames) {
359                 filesNonAutorized.clear();
360                 pendingFilesNo = (pendingFileNames == null) ? 0
361                         : pendingFileNames.size();
362
363                 authorizedFilesUploadedNo = 0;
364                 transfertNo++;
365                 initPendingFilesNo = pendingFilesNo;
366                 // allFilesSize = 0;
367                 fileList.clear();
368                 initFileList.clear();
369
370                 int error = 0;
371                 int allowedFileNo = 0;
372
373                 // Passe en revue un fichier après l'autre
374                 for (FileDetail f : pendingFileNames) {
375                     // Si l'extension n'est pas autorisée
376                     if (!allowedExtensions.contains(Utilities.getExtension(
377                             f.getFileName()).toLowerCase())
378                             && !allowedExtensions.contains("*")) {
379                         filesNonAutorized.add(f.getFileName());
380                         error = 2;
381
382                     } else {
383                         // Crée une barre de progression pour chaque fichier
384                         idFile++;
385                         allowedFileNo++;
386
387                         initFileList.add(new FileInfo(currentFileId
388                                 + allowedFileNo, f.getFileName(), f
389                                 .getContentLength(), f.getMimeType()));
390                         allFilesSize += f.getContentLength();
391                     }
392                     System.out.println("filename : " + f.getFileName() + ";"
393                             + "idFile : " + idFile + ";" + "allowedFileNo : "
394                             + allowedFileNo + ";" + "currentFileId : "
395                             + currentFileId + ";" + "assigné : "
396                             + (currentFileId + allowedFileNo));
397
398                 }
399
400                 // Si la taille max est plus grande que la taille max autorisée,
401                 // on envoie une erreur
402                 if (allFilesSize > maxQueueSize) {
403                     error = 3;
404                     idFile = 0;
405                     allowedFileNo = 0;
406                     for (FileDetail f : pendingFileNames) {
407                         filesNonAutorized.add(f.getFileName());
408                     }
409                 } else {
410                     // Désactivation de la zone Drag&Drop
411                     if (initFileList.size() > 0) {
412                         setEnabledUploadDropZone(false);
413                         setEnableUploadButton(false);
414                     }
415                 }
416
```



```
417                 System.out.println("---- initFileList -------");
418                 for (FileInfo initFl : initFileList) {
419                     System.out.println(initFl.getName());
420                 }
421
422                 notifyOnSelectedFiles(error);
423             }
424         };
425         upload.setHandler(handler);
426         upload.setButtonCaption(getUploadButtonCaption());
427         uploads.addComponent(upload);
428
429     }
430
431     /**
432      * Création d'une barre de progression
433      *
434      * @return (ProgressIndicator) Barre de progression
435      */
436     public ProgressIndicator createProgressIndicator() {
437         ProgressIndicator progressIndicator = new ProgressIndicator();
438         progressIndicator.setPollingInterval(300);
439         progressIndicator.setValue(0);
440         return progressIndicator;
441     }
442
443     /**
444      * Active/Désactive la zone de Drag/Drop
445      *
446      * @param enable
447      *            (true) Activé (false) Désactivé
448      */
449     public void setEnabledUploadDropZone(boolean enable) {
450         if (supportsFileDrops())
451             dropZone.setEnabled(enable);
452     }
453
454     /**
455      * Active/Désactive le bouton pour choisir les fichiers
456      *
457      * @param enable
458      *            (true) Activé (false) Désactivé
459      */
460     public void setEnableUploadButton(boolean enable) {
461         upload.setEnabled(enable);
462     }
463
464     /**
465      * Fixe les extensions de fichiers qui peuvent être envoyés
466      *
467      * @param extentions
468      *            Extensions de fichiers séparés par des virgules (exe,jpg,txt)
469      *            ou * pour tous les fichiers
470      */
471     public void setAllowedExtentions(String extentions) {
472         String[] ext = extentions.split(",");
473         allowedExtensions.clear();
474         for (String in : ext) {
475             allowedExtensions.add(in.trim());
476         }
477     }
478
```

```java
    /**
     * Retourne la liste des fichiers sélectionnés au départ
     *
     * @return Liste des fichiers sélectionnés
     */
    public ArrayList<FileInfo> getInitListFile() {
        return initFileList;
    }

    /**
     * Retourne la liste des extensions autorisées
     *
     * @return Extensions des fichiers autorisés
     */
    public ArrayList<String> getAllowedExtentions() {
        return allowedExtensions;
    }

    /**
     * Retourne la liste des fichiers non-autorisés
     *
     * @return Liste des fichiers non-autorisés
     */
    public ArrayList<String> getNonAutorizedFiles() {
        return filesNonAutorized;
    }

    /**
     * @return Nombre de fichié autorisés envoyés
     */
    public int getAllowedFilesUploaded() {
        return authorizedFilesUploadedNo;
    }

    /**
     * Fixe la taille limite pour le total des fichiers
     *
     * @param SizeLimit
     *            Taille maximum en octet
     */
    public void setMaxQueueSize(int SizeLimit) {
        maxQueueSize = SizeLimit;
    }

    /**
     * Retourne la taille des fichiers envoyés
     *
     * @return Taille des fichiers déjà sélectionnés
     */
    public long getQueueSize() {
        return allFilesSize;
    }

    /**
     * Soustrait une valeur à la taille de la liste des fichiers envoyés
     *
     * @param sizeToDecrease
     *            La valeur à soustraire
     */
    public void decreaseQueueSize(long sizeToDecrease) {
        this.allFilesSize = allFilesSize - sizeToDecrease;
        System.out
```



```java
                .println("Réduction de " + Utilities.formatSize(allFilesSize));
        System.out.println("Taille de la liste : " + allFilesSize + " ("
                + Utilities.formatSize(allFilesSize) + ")");
    }

    /**
     * Retourne le texte du bouton qui permet de sélectionner les fichiers
     *
     * @return Texte du bouton Parcourir...
     */
    public String getUploadButtonCaption() {
        return uploadButtonCaption;
    }

    /**
     * Fixe le texte du bouton Parcourir...
     *
     * @param uploadButtonCaption
     *            Texte du bouton
     */
    public void setUploadButtonCaption(String uploadButtonCaption) {
        this.uploadButtonCaption = uploadButtonCaption;
        Iterator<Component> componentIterator = uploads.getComponentIterator();
        while (componentIterator.hasNext()) {
            Component next = componentIterator.next();
            if (next instanceof MultiUpload) {
                MultiUpload upload = (MultiUpload) next;
                if (upload.isVisible()) {
                    upload.setButtonCaption(getUploadButtonCaption());
                }
            }
        }
    }

    private FileFactory fileFactory;

    public FileFactory getFileFactory() {
        if (fileFactory == null) {
            fileFactory = new TempFileFactory2();
        }
        return fileFactory;
    }

    public void setFileFactory(FileFactory fileFactory) {
        this.fileFactory = fileFactory;
    }

    protected FileBuffer createReceiver() {
        FileBuffer receiver = new FileBuffer(FieldType.FILE) {
            @Override
            public FileFactory getFileFactory() {
                return MultipleFileUpload.this.getFileFactory();
            }
        };
        return receiver;
    }

    /**
     * Renvoi l'interval de mise à jour de la barre de progression
     *
     * @return Interval de mise à jour de la ProgressBar
     */
```

```
603     protected int getPollinInterval() {
604         return 500;
605     }
606 
607     /**
608      * Retourne le nombre de fichiers en attente
609      *
610      * @return Nombre de fichiers en attente de téléchargement
611      */
612     protected int getPendingFiles() {
613         return pendingFilesNo;
614     }
615 
616     @Override
617     public void attach() {
618         super.attach();
619         if (supportsFileDrops()) {
620             prepareDropZone();
621         }
622     }
623 
624     // ------------------------------------------------------------------------
    --------
625     // ------------------------------ HTML5 Drag&Drop
626     // ------------------------------------
627     // ------------------------------------------------------------------------
    --------
628 
629     private DragAndDropWrapper dropZone;
630 
631     /** Indique si la dropZone doit être visible ou non */
632     private boolean dropZoneVisible = true;
633 
634     /**
635      * Retourne la visibilité de la dropZone.
636      *
637      * @return true : dropZone visible, false : dropZone masquée.
638      */
639     public boolean isDropZoneVisible() {
640         return dropZoneVisible;
641     }
642 
643     /**
644      * Définit la visibilité de la dropZone
645      *
646      * @param Définit
647      *            l'attribut dropZoneVisible à la valeur spécifiée
648      */
649     public void setDropZoneVisible(boolean dropZoneVisible) {
650         if (supportsFileDrops()) {
651             this.dropZoneVisible = dropZoneVisible;
652 
653             if (!dropZoneVisible)
654                 dropZone.setStyleName("v-multifileupload-dropzone-hide");
655             else
656                 dropZone.removeStyleName("v-multifileupload-dropzone-hide");
657         }
658     }
659 
660     /**
661      * Indique si un envoi est en cours ou s'il y a encore des fichiers en
662      * attentes
```



```
663      *
664      * @return true si un fichier est encours de téléchargement ou s'il y a
665      *         encore des fichiers en attente
666      */
667     public boolean isInProcess() {
668         return (isInProcess || pendingFilesNo > 0);
669     }
670 
671     /**
672      * Règle le DragAndDropWrapper pour accepter les dépôts de fichiers
673      * multiples dans la fenêtre.
674      */
675     private void prepareDropZone() {
676         if (dropZone == null && isDropZoneVisible()) {
677             Component label = new Label(getAreaText(), Label.CONTENT_XHTML);
678             label.setSizeUndefined();
679 
680             dropZone = new DragAndDropWrapper(label);
681             dropZone.setStyleName("v-multifileupload-dropzone");
682             dropZone.setSizeFull();
683             addComponent(dropZone, 2);
684             dropZone.setDropHandler(this);
685             addStyleName("no-horizontal-drag-hints");
686             addStyleName("no-vertical-drag-hints");
687 
688         }
689     }
690 
691     /**
692      * Retourne le texte de la zone de drag&drop
693      *
694      * @return Texte de la zone de drag&drop
695      */
696     public String getAreaText() {
697         return areaText;
698     }
699 
700     /**
701      * Définit le texte de la zone de drag&drop
702      *
703      * @param areaText
704      *            Texte de la zone de drag&drop. Peut contenir du code HTML
705      */
706     public void setAreaText(String areaText) {
707         this.areaText = areaText;
708     }
709 
710     /**
711      * Indique si le drag&drop est supporté par le navigateur ou non.<br>
712      * <br>
713      * Les navigateurs qui prennent en charge cette fonctionnalité sont :<br>
714      * <ul>
715      * <li>Chrome</li>
716      * <li>Firefox</li>
717      * <li>Safari</li>
718      * </ul>
719      *
720      * @return true : le drag&drop est activé.
721      */
722     protected boolean supportsFileDrops() {
723         AbstractWebApplicationContext context = (AbstractWebApplicationContext)
    getApplication()
```

```
                .getContext();
        WebBrowser browser = context.getBrowser();
        if (browser.isChrome()) {
            return true;
        } else if (browser.isFirefox()) {
            return true;
        } else if (browser.isSafari()) {
            return true;
        }
        return false;
    }

    /**
     * Méthode appelée obligatoirement à la fin de chaque envoi de fichier
     *
     * @param file
     *            Informations sur le fichier envoyé
     * @param fileName
     *            Nom du fichier envoyé
     * @param mimeType
     *            Type MIME du fichier envoyé
     * @param length
     *            Taille du fichier envoyé
     */
    abstract protected void handleFile(File file, String fileName,
            String mimeType, long length);

    /**
     * Une méthode d'assistance pour définir DirectoryFileFactory avec le chemin
     * du répertoire spécifié
     *
     * @param directoryWhereToUpload
     *            Répertoire dans lequel envoyer les fichiers
     */
    public void setRootDirectory(String directoryWhereToUpload) {
        setFileFactory(new DirectoryFileFactory(
                new File(directoryWhereToUpload)));
    }

    /**
     * Retourne les critères d'acceptation d'un fichier pour l'envoi Ne pas
     * utiliser cette méthode pour le moment, elle n'est pas encore implémentée
     */
    public AcceptCriterion getAcceptCriterion() {
        return AcceptAll.get();
    }

    /**
     * Méthode exécutée lorsqu'un fichier est déposé dans la zone prévue à cet
     * effet
     */
    public void drop(DragAndDropEvent event) {
        DragAndDropWrapper.WrapperTransferable transferable = (WrapperTransferable)
event
                .getTransferable();
        Html5File[] files = transferable.getFiles();

        filesNonAutorized.clear();
        pendingFilesNo = (files == null) ? 0 : files.length;

        int error = 0;
```



```
        if (files != null) {
            authorizedFilesUploadedNo = 0;
            transfertNo++;
            initPendingFilesNo = pendingFilesNo;
            allFilesSize = 0;
            fileList.clear();
            initFileList.clear();

            // Calcul de la taille total des fichiers
            for (final Html5File html5File : files) {
                allFilesSize += html5File.getFileSize();
            }

            // Si la taille total des fichiers dépasse la taille limite, on ne
            // laisse pas envoyer
            if (allFilesSize <= maxQueueSize) {
                setEnableUploadButton(false);

                int allowedFileNo = 0;
                for (final Html5File html5File : files) {
                    // Si l'extension n'est pas autorisée
                    if (!allowedExtensions.contains(Utilities.getExtension(
                            html5File.getFileName()).toLowerCase())
                            && !allowedExtensions.contains("*")) {
                        // On décrémente le nombre de fichiers dans la liste
                        // d'attente pour chaque fichier non autorisé
                        initPendingFilesNo--;
                        pendingFilesNo--;
                        // On crée un tableau contenant les fichiers non
                        // autorisés
                        filesNonAutorized.add(html5File.getFileName());

                        error = 2;
                    } else {
                        idFile++;
                        allowedFileNo++;

                        initFileList.add(new FileInfo(currentFileId
                                + allowedFileNo, html5File.getFileName(),
                                html5File.getFileSize(), html5File.getType()));
                        final FileBuffer receiver = createReceiver();
                        html5File.setStreamVariable(new StreamVariable() {

                            private String name;
                            private String mime;

                            public OutputStream getOutputStream() {
                                return receiver.receiveUpload(name, mime);
                            }

                            public boolean listenProgress() {
                                return true;
                            }

                            public void onProgress(StreamingProgressEvent event) {
                                float p = (float) event.getBytesReceived()
                                        / (float) event.getContentLength();

                                progression = p * 100;
                                notifyUploadProgess(currentFileId,
                                        event.getFileName(), pendingFilesNo,
                                        progression);
```

```
847                     }
848
849                     public void streamingStarted(
850                             StreamingStartEvent event) {
851                         name = event.getFileName();
852                         mime = event.getMimeType();
853
854                         pendingFilesNo--;
855                         currentFileId++;
856
857                         notifyUploadStart(currentFileId,
858                                 event.getFileName(), pendingFilesNo);
859                     }
860
861                     public void streamingFinished(
862                             StreamingEndEvent event) {
863                         authorizedFilesUploadedNo++;
864
865                         String originalFileName = event.getFileName();
866                         File originalFile = new File(Global.UPLOAD_DIR
867                                 + originalFileName);
868
869                         File newFile = null;
870                         // Renommage du fichier
871                         if (originalFile.exists()) {
872                             originalFile.setWritable(true);
873                             originalFile.setReadable(true);
874                             originalFile.setExecutable(true);
875
876                             String newName = Global.UPLOAD_DIR
877                                     + Utilities
878                                             .getNewFileName(originalFileNam
    e);
879                             // getWindow().showNotification("Enregistré
    sous",
880                             // newName);
881                             newFile = new File(newName);
882
883                             originalFile.renameTo(newFile);
884                         }
885
886                         fileList.add(newFile);
887                         notifyUploadFinished(currentFileId,
888                                 event.getFileName(), newFile,
889                                 pendingFilesNo);
890
891                         if (pendingFilesNo == 0) {
892                             notifyUploadComplete(fileList);
893                             setEnableUploadButton(true);
894                         }
895
896                         handleFile(newFile, newFile.getName(),
897                                 html5File.getType(),
898                                 html5File.getFileSize());
899
900                         receiver.setValue(null);
901
902                     }
903
904                     public void streamingFailed(
905                             StreamingErrorEvent event) {
906                         // progressBars.removeComponent(pi);
```



```
907                             notifyUploadError(event.getFileName(),
908                                     pendingFilesNo);
909                         }
910
911                         public boolean isInterrupted() {
912                             return false;
913                         }
914                     });
915                 }
916
917             }
918
919             // Si l'utilisateur n'a ajouté que des fichiers qui ne sont pas
920             // autorisés, on réactive l'upload
921             if (filesNonAutorized.size() > 0 && initPendingFilesNo == 0) {
922                 setEnableUploadButton(true);
923             }
924         } else {
925             error = 4;
926         }
927     } else {
928         error = 1;
929     }
930
931     System.out.println(allFilesSize + " Octets -- "
932             + Utilities.formatSize(allFilesSize));
933
934     // Exécuté lorsque les fichiers sont ajoutés dans la zone
935     notifyOnSelectedFiles(error);
936 }
937
938 }
939
```

TempFileFactory2.java

```java
package common.component.upload;

import java.io.File;

/**
 * Enregistrement temporaire des fichiers sélectionnés pour l'upload.
 */
class TempFileFactory2 implements FileFactory {
    /**
     * Surcharge de la méthode createFile de la class FileFactory.
     */
    public File createFile(String fileName, String mimeType) {
        final String tempFileName = "upload_tmpfile_"
                + System.currentTimeMillis();
        try {
            System.out.println(File.createTempFile(tempFileName, null)
                    .getName());
            return File.createTempFile(tempFileName, null);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

}
```


UploadActionListener.java

```java
package common.component.upload;

import java.io.File;

/**
 * Interface contenant les évènements reçus par les fichiers lors de l'envoi.<br>
 * Son implémentation permet de détecter le début et la fin d'un transfert, de
 * détecter les erreurs et de suivre la progression des fichiers.
 *
 * @author steffen.c-on, Dominique Roduit
 * @version 18.03.2013
 */
public interface UploadActionListener {
    /**
     * Called if a upload of a file is started.
     *
     * @param filename
     *            name of the file which is currently uploading
     * @param pendingFiles
     *            number of pending files (without the currently uploading file)
     */
    void fileUploadStarted(int idFile, String filename, int pendingFiles);

    /**
     * Called if a upload of a file is finished successful.
     *
     * @param filename
     *            name of the file which is currently finished
     * @param pendingFiles
     *            number of pending files
     */
    void fileUploadFinished(int idFile, String filename, File file,
            int pendingFiles);

    /**
     * Called if a upload of a file is finished with an error.
     *
     * @param filename
     *            name of the file which is aborted
     * @param pendingFiles
     *            number of pending files
     */
    void fileUploadError(String filename, int pendingFiles);

    /**
     * Appelé durant la progression de l'upload d'un fichier
     *
     * @param idFile
     *            Identifiant du fichier
     * @param filename
     *            Nom du fichier
     * @param pendingFiles
     *            Nombre de fichiers en attentes
     * @param progress
     *            Progression (de 0 à 1)
     */
    void fileUploadProgress(int idFile, String filename, int pendingFiles,
            float progress);

    /**
     * Appelé lorsque tout les fichiers sont envoyés avec succès
     *
```

```
66      * @param fileList
67      *            Liste des fichiers envoyés
68      */
69     void fileUploadComplete(ArrayList<File> fileList);
70
71     /**
72      * Appelé lorsque les fichiers sont sélectionnés (ou déposés dans la
73      * fenêtre)
74      *
75      * @param error
76      *            Code d'erreur retourné (s'il y en a une)
77      */
78     void onSelectedFiles(int error);
79
80 }
```

PACKAGE

FORM

- ContactForm
- WizFoldDest
- WizFoldNew
- WizFoldUpload

Formulaires de gestion, assistants de création des dossiers, fenêtres flottantes

```java
package form;

import global.Global;

/**
 * Cette class permet l'ajout et la mise à jour des contacts via un formulaire
 * affiché dans une sous-fenêtre. Un constructeur permet l'ajout, un autre
 * permet la mise à jour.
 *
 * @author Dominique
 *
 */
public class ContactForm extends CustomComponent {

    /**
     * Contient toutes les instances qui doivent être accessibles dans toute
     * l'application
     **/
    private GlobalObjects global = CccvsTransfert.getGlobalMethod();
    /** Instance de la sous-fenêtre qui va contenir ce formulaire **/
    private Window winContact = global.getWindowContact();
    /** Instance de la table pour l'affichage des contacts **/
    private ContactTable tblContact = global.getTableContact();
    /** Instance du bouton d'ajout d'un contact **/
    private Button btAddContact = global.getBtAddContact();
    /** Contient toutes les requêtes SQL **/
    private tbl_contacts queries = new tbl_contacts();

    /** Layout vertical permettant d'afficher le formulaire en ligne **/
    private VerticalLayout mainLayout;
    /** Modèle de contact **/
    private Contact contact;
    /** Bouton Ajouter **/
    private Button btApply;
    /** Formulaire permettant l'ajout et l'édition **/
    private Form form;
    /** Bas du formulaire **/
    private HorizontalLayout footerLayout;
    /** Zone contenant les boutons du formulaire **/
    private HorizontalLayout buttons;
    /** Bouton Annuler **/
    private Button cancel;
    /** Etabli le lien entre le modèle de Contact et les éléments du formulaire **/
    private BeanItem<Contact> contactItem;

    /**
     * Affiche un formulaire pour l'ajout de contact
     */
    public ContactForm() {
        doForm();
        contact.setPK(0);
    }

    /**
     * Affiche un formulaire pour l'édition d'un contact
     *
     * @param PK
     *            La clé primaire du contact à éditer
     */
    public ContactForm(String PK) {
        // Renommage de la fenêtre
        winContact.setCaption(Global.i("CAPTION_CONTACT_EDIT"));
```



```java

        doForm();

        btApply.setCaption(Global.i("CAPTION_SAVE"));
        contact.setPK(Integer.parseInt(PK));
        try {
            ResultSet c_data = Sql.query(queries.getSelectByPKContactQuery(PK));
            c_data.first();
            contact.setMail(c_data.getString("contact_mail"));
            contact.setName(c_data.getString("contact_name"));
            form.discard();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    /**
     * Construction du formulaire
     */
    private void doForm() {
        mainLayout = new VerticalLayout();
        mainLayout.setSizeFull();

        contact = new Contact();
        contactItem = new BeanItem<Contact>(contact);

        form = new Form();
        form.setWidth("300px");

        form.setFormFieldFactory(new ContactFieldFactory());
        form.setItemDataSource(contactItem);

        form.setVisibleItemProperties(Arrays.asList(new String[] { "name",
                "mail" }));

        mainLayout.addComponent(form);
        mainLayout.setComponentAlignment(form, Alignment.MIDDLE_CENTER);

        footerLayout = new HorizontalLayout();
        footerLayout.setWidth("100%");

        buttons = new HorizontalLayout();
        buttons.setSpacing(true);

        // Création du bouton "Annuler"
        cancel = new Button(Global.i("CAPTION_CANCEL"),
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        form.discard();
                        (winContact.getParent()).removeWindow(winContact);
                    }
                });
        cancel.setTabIndex(4);
        buttons.addComponent(cancel);
        buttons.setComponentAlignment(cancel, Alignment.MIDDLE_CENTER);

        // Création du bouton "Ajouter"
        btApply = new Button(Global.i("CAPTION_ADD"),
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        try {
                            form.commit();
```

```
                    } catch (Exception e) {
                        System.out.println(e);
                    }
                }
            });
        btApply.setTabIndex(3);
        buttons.addComponent(btApply);

        footerLayout.addComponent(buttons);
        footerLayout.setComponentAlignment(buttons, Alignment.TOP_CENTER);
        form.setFooter(footerLayout);

        btApply.setClickShortcut(KeyCode.ENTER);
        btApply.addListener(SubmitListener);

        form.focus();

        setCompositionRoot(mainLayout);
    }

    /**
     * Retourne une instance du bouton "Ajouter" de la sous-fenêtre
     *
     * @return Instance du bouton "Ajouter"
     */
    public Button getButtonApply() {
        return btApply;
    }

    /**
     * Ecouteur exécuté lors de la validation du formulaire
     */
    private ClickListener SubmitListener = new ClickListener() {
        @Override
        public void buttonClick(ClickEvent event) {
            if (form.isValid()) {
                // On ne laisse aux externe ajouter que des adresses de
                // personnes internes à la caisse
                if (UserSession.isInternal()
                        || (!UserSession.isInternal() && Utilities
                                .isInternal(contact.getMail()))) {
                    // Si le modèle de contact ne contient pas de valeur
                    if (contact.getPK() == 0) { // On ajoute le contact
                        Sql.exec(queries.getInsertContactQuery(
                                contact.getName(), contact.getMail()));
                    } else { // On le met à jour
                        Sql.exec(queries.getUpdateContactQuery(
                                Integer.toString(contact.getPK()),
                                contact.getName(), contact.getMail()));
                    }

                    if (tbl_contacts.getNumberOfContact() == 1) {
                        CccvsTransfert.loadModule(new Module(ContactModule
                                .getBodyRibbonContent(), ContactModule
                                .getBodyContent()));
                    }

                    // Mise à jour de la table
                    tblContact.reload();
                    // Mise à jour de la table du menu
                    MenuContactTable menuContactTable = global
                            .getMenuContactTable();
```



```
                    menuContactTable.reload();

                    // Fermeture de la sous-fenêtre
                    (winContact.getParent()).removeWindow(winContact);
                    // Focus sur le bouton d'ajout de contact
                    btAddContact.focus();

                    // On redonne le focus a la ligne en cours ou à la dernière
                    // ligne ajoutée
                    if (contact.getPK() != 0) {
                        tblContact.select(contact.getPK());
                    } else {
                        try {
                            ResultSet lastPK = Sql.query(tbl_contacts
                                    .getLastContact());
                            lastPK.first();
                            tblContact.select(lastPK.getInt("LAST"));
                        } catch (SQLException e) {
                            e.printStackTrace();
                        } finally {
                            Sql.Disconnect();
                        }
                    }
                } else {
                    getWindow().showNotification(
                            null,
                            Global.i("CAPTION_RESTRICT_EMAIL") + " "
                                    + Global.getParm("EMAIL_INTERNE_SUFFIXE"),
                            Notification.TYPE_WARNING_MESSAGE);
                }
            }
        }
    };

    /**
     * Classe liées au formulaire de la sous-fenêtre. Elle permet de formatter
     * et de programmer le comportement des éléments du formulaire.
     *
     * @author Dominique
     *
     */
    private class ContactFieldFactory extends DefaultFieldFactory {
        /**
         * Création des éléments du formulaire
         */
        public Field createField(Item item, Object propertyId,
                Component uiContext) {
            Field f;

            f = super.createField(item, propertyId, uiContext);

            if (propertyId.equals("name")) {
                TextField tf = (TextField) f;
                tf.setRequired(true);
                tf.setCaption(Global.i("CAPTION_NAME"));
                tf.setRequiredError(Global.i("CAPTION_EMPTY_NAME"));
                tf.setColumns(18);
                tf.setTabIndex(1);
                tf.addValidator(new StringLengthValidator(Global
                        .i("CAPTION_LENGTH_RESTRICT_1"), 3, 25, false));
            } else if (propertyId.equals("mail")) {
                TextField tf = (TextField) f;
```

```
293                 tf.setRequired(true);
294                 tf.setCaption(Global.i("CAPTION_EMAIL"));
295                 tf.setRequiredError(Global.i("CAPTION_EMPTY_MAIL"));
296                 tf.setColumns(18);
297                 tf.setTabIndex(2);
298                 tf.addValidator(new EmailValidator(Global
299                         .i("CAPTION_ERROR_MAIL")));
300             } else if (propertyId.equals("PK")) {
301                 TextField tf = (TextField) f;
302             }
303             return f;
304         }
305     }
306 }
307
```



```
1 package form;
2
3 import java.sql.ResultSet;
56
57 /**
58  * 2e vue de l'assistant pour la création d'un dossier.<br>
59  * Cette vue affiche un tableau ainsi qu'une liste déroulante contenant la liste
60  * de tous les contacts de l'utilisateur. L'utilisateur peut choisir lesquels
61  * contacts il souhaite ajouter comme destinataires. Cette class transmet les
62  * informations sélectionnées à la vue suivante ({@link WizFoldUpload}).
63  *
64  * @author Dominique Roduit
65  *
66  */
67 public class WizFoldDest extends CustomComponent {
68     /**
69      * Contient toutes les instances qui doivent être accessibles dans toute
70      * l'application
71      **/
72     private GlobalObjects global = CccvsTransfert.getGlobalMethod();
73     /** Instance de la sous-fenêtre qui va contenir ce formulaire **/
74     private Window winFolder = global.getWindowFolder();
75     /** Layout vertical permettant d'afficher le formulaire en ligne **/
76     private VerticalLayout mainLayout;
77     /** Modèle de contact **/
78     private Contact contact;
79     /** Bouton Ajouter **/
80     private Button btApply;
81     /** Formulaire permettant l'ajout et l'édition **/
82     private Form form;
83     /** Bas du formulaire **/
84     private HorizontalLayout footerLayout;
85     /** Zone contenant les boutons du formulaire **/
86     private HorizontalLayout buttons;
87     /** Bouton Annuler **/
88     private Button cancel;
89     /** Etabli le lien entre le modèle de Contact et les éléments du formulaire **/
90     private BeanItemContainer<Contact> container = new BeanItemContainer<Contact>(
91             Contact.class);
92     /** Informations sur le dossier **/
93     private Folder folder = null;
94     /** Liste déroulante qui contient les contacts **/
95     private ComboBox cbxContact;
96     /** Tableau visuel contenant les destinataires sélectionnés **/
97     private Table tblRecipients;
98     /** Tableau contenant les destinataires sélectionnés **/
99     private ArrayList<Integer> slctContacts = new ArrayList<Integer>();
100    /** Action "Supprimer" du menu contextuel **/
101    private Action ACTION_DEL;
102
103    /**
104     * Affiche un formulaire pour l'ajout de destinataires à un dossier
105     *
106     * @param folder
107     *            Informations sur le dossier, définies dans le wizard 1
108     */
109    public WizFoldDest(Folder folder) {
110        this.folder = folder;
111
112        ACTION_DEL = new Action(Global.i("CAPTION_REVOKE"), new ThemeResource(
113                Global.PATH_THEME_RESSOURCES + "false.png"));
114
```

```
115         winFolder.setClosable(false);
116         winFolder.setWidth("540px");
117         winFolder.setHeight("300px");
118         winFolder.setCaption(Global.i("CAPTION_TRANSFERT") + " - "
119                 + Global.i("TITLE_WINDOW_ADD_RECIPIENTS"));
120         winFolder.setIcon(new ThemeResource(Global.PATH_THEME_RESSOURCES
121                 + "user--arrow.png"));
122
123         doForm();
124     }
125
126     /**
127      * Construction du formulaire
128      */
129     private void doForm() {
130         mainLayout = new VerticalLayout();
131         mainLayout.setSizeFull();
132         mainLayout.setSpacing(true);
133
134         contact = new Contact();
135
136         // Ajout d'une combobox qui permet de sélectionner les contacts a
137         // ajouter
138         ArrayList<Contact> contactList = getContactList();
139         container.addAll(contactList);
140
141         cbxContact = new ComboBox(null);
142         cbxContact.setContainerDataSource(container);
143         cbxContact.setNullSelectionAllowed(false);
144         cbxContact.setImmediate(true);
145         cbxContact.setItemCaptionPropertyId("name");
146         cbxContact
147                 .setItemCaptionMode(AbstractSelect.ITEM_CAPTION_MODE_PROPERTY);
148         cbxContact.setWidth("100%");
149         cbxContact.setInputPrompt(Global.i("CAPTION_SELECT_CONTACT"));
150         cbxContact.focus();
151
152         tblRecipients = new Table();
153         tblRecipients.setSizeFull();
154         tblRecipients.setHeight("160px");
155         tblRecipients.setSelectable(true);
156         tblRecipients.setColumnCollapsingAllowed(true);
157         tblRecipients.addContainerProperty("name", String.class, null,
158                 Global.i("CAPTION_NAME"), null, null);
159         tblRecipients.addContainerProperty("email", String.class, null,
160                 Global.i("CAPTION_ADRESSE_EMAIL"), null, null);
161         tblRecipients.addContainerProperty("internal", String.class, null,
162                 Global.i("CAPTION_INTERNAL"), null, Table.ALIGN_CENTER);
163         tblRecipients.setColumnCollapsed("internal", true);
164         tblRecipients.addActionHandler(new Action.Handler() {
165             public void handleAction(Action action, Object sender, Object target) {
166                 if (action == ACTION_DEL) {
167                     tblRecipients.removeItem(tblRecipients.getValue());
168                 }
169             }
170
171             public Action[] getActions(Object target, Object sender) {
172                 if (target != null) {
173                     return new Action[] { ACTION_DEL };
174                 } else {
175                     return null;
176                 }
```



```
177             }
178         });
179         tblRecipients.addListener(new ItemClickListener() {
180             public void itemClick(ItemClickEvent event) {
181                 if (event.getButton() == ItemClickEvent.BUTTON_RIGHT) {
182                     tblRecipients.setValue(null); // Dé-sélectionne tout
183                     tblRecipients.select(event.getItemId()); // Sélectionne la
184                                                              // ligne en
185                                                              // cours
186                 }
187             }
188         });
189
190         cbxContact.addListener(new Property.ValueChangeListener() {
191             public void valueChange(ValueChangeEvent event) {
192                 if (cbxContact.getValue() != null) {
193                     Contact slctContact = ((Contact) cbxContact.getValue());
194
195                     String interne_caption = (slctContact.isInternal()) ? Global
196                             .i("CAPTION_INTERNAL") : Global
197                             .i("CAPTION_EXTERNAL");
198                     Object[] item = new Object[] { slctContact.getName(),
199                             slctContact.getMail(), interne_caption };
200                     tblRecipients.addItem(item, slctContact.getPK());
201                     cbxContact.setValue(null);
202                 }
203             }
204         });
205
206         mainLayout.addComponent(cbxContact);
207         mainLayout.addComponent(tblRecipients);
208
209         buttons = new HorizontalLayout();
210         buttons.setSpacing(true);
211         mainLayout.addComponent(buttons);
212         mainLayout.setComponentAlignment(buttons, Alignment.BOTTOM_RIGHT);
213
214         cancel = new Button(Global.i("CAPTION_CANCEL"),
215                 new Button.ClickListener() {
216                     public void buttonClick(ClickEvent event) {
217                         (winFolder.getParent()).removeWindow(winFolder);
218                     }
219                 });
220         cancel.setTabIndex(4);
221         buttons.addComponent(cancel);
222         buttons.setComponentAlignment(cancel, Alignment.BOTTOM_RIGHT);
223
224         // Création du bouton "Ajouter"
225         btApply = new Button(Global.i("CAPTION_NEXT") + " »");
226         btApply.setTabIndex(3);
227         buttons.addComponent(btApply);
228
229         btApply.setClickShortcut(KeyCode.ENTER);
230         btApply.addListener(SubmitListener);
231
232         setCompositionRoot(mainLayout);
233     }
234
235     /**
236      * Retourne la liste des contacts de l'utilisateur connecté
237      *
238      * @return Liste des contacts
```

```
239     */
240    private ArrayList<Contact> getContactList() {
241        ArrayList<Contact> contacts = new ArrayList<Contact>();
242        ResultSet data = Sql.query(tbl_contacts.getSelectAllContactQuery());
243        try {
244            while (data.next()) {
245                contacts.add(new Contact(data.getInt("PKNoContact"), data
246                        .getString("contact_name"), data
247                        .getString("contact_mail"), data
248                        .getBoolean("contact_internal")));
249            }
250        } catch (SQLException e) {
251            e.printStackTrace();
252        }
253        return contacts;
254    }
255
256    /**
257     * Action lors de la soumission du wizard
258     */
259    private ClickListener SubmitListener = new ClickListener() {
260        @Override
261        public void buttonClick(ClickEvent event) {
262            // On récupère les contact sélectionnés
263            for (Object id : tblRecipients.getItemIds()) {
264                slctContacts.add(Integer.parseInt(id.toString()));
265            }
266
267            // Si on a au moins un contact dans notre tableau
268            if (slctContacts.size() > 0) {
269                // Passage au wizard suivant
270                winFolder.removeAllComponents();
271                winFolder.addComponent(new WizFoldUpload(folder, slctContacts));
272            } else {
273                getWindow().showNotification(null,
274                        Global.i("CAPTION_ADD_RECIPIENTS"),
275                        Notification.TYPE_WARNING_MESSAGE);
276            }
277        }
278    };
279
280}
281
```



```
1 package form;
2
3 import java.sql.ResultSet;
49
50 /**
51  * Premier Wizard de l'assistant de création d'un dossier.<br>
52  * Il permet la définition du nom du dossier, de sa description, et le choix de
53  * sa durée de vie.<br>
54  * Aucune information n'est enregistrée dans ce Wizard. Toutes les données
55  * saisies par l'utilisateur sont transmises aux vues suivantes (
56  * {@link WizFoldDest}, {@link WizFoldUpload}).
57  *
58  * @author Dominique Roduit
59  *
60  */
61 public class WizFoldNew extends CustomComponent {
62     /**
63      * Contient toutes les instances qui doivent être accessibles dans toute
64      * l'application
65      **/
66     private GlobalObjects global = CccvsTransfert.getGlobalMethod();
67     /** Instance de la sous-fenêtre qui va contenir ce formulaire **/
68     private Window winFolder = global.getWindowFolder();
69     /** Layout vertical permettant d'afficher le formulaire en ligne **/
70     private VerticalLayout mainLayout;
71     /** Modèle de contact **/
72     private Folder folder;
73     /** Bouton Ajouter **/
74     private Button btApply;
75     /** Formulaire permettant l'ajout et l'édition **/
76     private Form form;
77     /** Bas du formulaire **/
78     private HorizontalLayout footerLayout;
79     /** Zone contenant les boutons du formulaire **/
80     private HorizontalLayout buttons;
81     /** Bouton Annuler **/
82     private Button cancel;
83     /** Etabli le lien entre le modèle de Contact et les éléments du formulaire **/
84     private BeanItem<Folder> contactItem;
85
86     /**
87      * Affiche un formulaire pour l'ajout de contact
88      */
89     public WizFoldNew() {
90         doForm();
91     }
92
93     /**
94      * Construction du formulaire
95      */
96     private void doForm() {
97         mainLayout = new VerticalLayout();
98         mainLayout.setSizeFull();
99
100        folder = new Folder();
101        contactItem = new BeanItem<Folder>(folder);
102
103        form = new Form();
104        form.setWidth("360px");
105
106        form.setFormFieldFactory(new TransfertFieldFactory());
107        form.setItemDataSource(contactItem);
```

```
108
109         form.setVisibleItemProperties(Arrays.asList(new String[] { "name",
110                 "expiration", "description" }));
111
112         mainLayout.addComponent(form);
113         mainLayout.setComponentAlignment(form, Alignment.MIDDLE_CENTER);
114
115         footerLayout = new HorizontalLayout();
116         footerLayout.setWidth("100%");
117
118         buttons = new HorizontalLayout();
119         buttons.setSpacing(true);
120
121         // Création du bouton "Annuler"
122         cancel = new Button(Global.i("CAPTION_CANCEL"),
123                 new Button.ClickListener() {
124                     public void buttonClick(ClickEvent event) {
125                         form.discard();
126                         (winFolder.getParent()).removeWindow(winFolder);
127                     }
128                 });
129         cancel.setTabIndex(4);
130         buttons.addComponent(cancel);
131         buttons.setComponentAlignment(cancel, Alignment.BOTTOM_RIGHT);
132
133         // Création du bouton "Suivant"
134         btApply = new Button(Global.i("CAPTION_NEXT") + " »",
135                 new Button.ClickListener() {
136                     public void buttonClick(ClickEvent event) {
137                         try {
138                             form.commit();
139                         } catch (Exception e) {
140                             System.out.println(e);
141                         }
142                     }
143
144                 });
145         btApply.setTabIndex(3);
146         buttons.addComponent(btApply);
147
148         footerLayout.addComponent(buttons);
149         footerLayout.setComponentAlignment(buttons, Alignment.BOTTOM_RIGHT);
150         form.setFooter(footerLayout);
151
152         // btApply.setClickShortcut(KeyCode.ENTER);
153         btApply.addListener(SubmitListener);
154
155         form.focus();
156
157         setCompositionRoot(mainLayout);
158     }
159
160     private ClickListener SubmitListener = new ClickListener() {
161         @Override
162         public void buttonClick(ClickEvent event) {
163             if (folder.getName().length() < 3 || folder.getName().length() > 50) {
164                 winFolder.setHeight("290px");
165             }
166
167             if (form.isValid()) {
168                 // Création du dossier
169                 System.out.println(tbl_folders.getInsertFolder(
```



```
170                         folder.getName(), folder.getExpiration(),
171                         folder.getDescription()));
172
173                 winFolder.removeAllComponents();
174                 // On passe au wizard suivant en passant en paramètre la PK du
175                 // dossier créé
176                 winFolder.addComponent(new WizFoldDest(folder));
177             }
178         }
179     };
180
181     /**
182      * Classe liées au formulaire de la sous-fenêtre. Elle permet de formatter
183      * et de programmer le comportement des éléments du formulaire.
184      *
185      * @author Dominique
186      *
187      */
188     private class TransfertFieldFactory extends DefaultFieldFactory {
189         private Slider sldExpiration = new Slider(null);
190         private TextArea descriptionArea = new TextArea();
191
192         public TransfertFieldFactory() {
193             sldExpiration.setWidth("96%");
194             sldExpiration.setMin(Integer.parseInt(Global
195                     .getParm("FOLDER_MIN_AVAILABILITY")));
196             sldExpiration.setMax(Integer.parseInt(Global
197                     .getParm("FOLDER_MAX_AVAILABILITY")));
198             sldExpiration.setImmediate(true);
199             sldExpiration.setCaption(Global.i("CAPTION_EXPIRATION_IN_DAYS"));
200             sldExpiration.setRequiredError(Global.i("CAPTION_EMPTY_MAIL"));
201
202             descriptionArea.setCaption(Global.i("CAPTION_DESCRIPTION"));
203             descriptionArea.setRows(3);
204             descriptionArea.setColumns(18);
205             descriptionArea.setTabIndex(2);
206         }
207
208         /**
209          * Création des éléments du formulaire
210          */
211         public Field createField(Item item, Object propertyId,
212                 Component uiContext) {
213             Field f;
214
215             if (propertyId.equals("expiration")) {
216                 return sldExpiration;
217             } else if (propertyId.equals("description")) {
218                 return descriptionArea;
219             } else {
220                 f = super.createField(item, propertyId, uiContext);
221             }
222
223             if (propertyId.equals("name")) {
224                 final TextField tf = (TextField) f;
225                 tf.setRequired(true);
226                 tf.setCaption(Global.i("CAPTION_FOLDER_NAME"));
227                 tf.setRequiredError(Global.i("CAPTION_EMPTY_NAME"));
228                 tf.setColumns(18);
229                 tf.setTabIndex(1);
230                 tf.addValidator(new StringLengthValidator(Global
231                         .i("CAPTION_LENGTH_RESTRICT_2"), 3, 50, false));
```

```
232                 tf.addShortcutListener(new ShortcutListener("enter",
233                         ShortcutAction.KeyCode.ENTER, null) {
234                     public void handleAction(Object sender, Object target) {
235                         if (target == tf) {
236                             if (tf.getValue().toString().length() < 3
237                                     || tf.getValue().toString().length() > 50) {
238                                 winFolder.setHeight("290px");
239                             }
240                             btApply.click();
241                         }
242                     }
243                 });
244             }
245 
246             return f;
247         }
248     }
249 }
250 
```



```
1 package form;
2 
3 import java.io.File;
77 
78 /**
79  * 3e vue de l'assistant de création d'un dossier.<br>
80  * Ce Wizard permet l'upload des fichiers, le stockage dans un tableau et le
81  * renommage des fichiers envoyés.<br>
82  * Sur le clic du bouton "Terminer", toutes les informations sont enregistrées
83  * dans la base de données, les mails envoyés aux destinataires et l'archive
84  * contenant l'ensemble des fichiers du dossier est générée.
85  *
86  * @author Dominique Roduit
87  *
88  */
89 public class WizFoldUpload extends CustomComponent {
90     /**
91      * Contient toutes les instances qui doivent être accessibles dans toute
92      * l'application
93      **/
94     private GlobalObjects global = CccvsTransfert.getGlobalMethod();
95     /** Instance de la sous-fenêtre qui va contenir ce formulaire **/
96     private Window winFolder = global.getWindowFolder();
97     /** Layout vertical permettant d'afficher le formulaire en ligne **/
98     private VerticalLayout mainLayout;
99     /** Bouton Ajouter **/
100    private static Button btFinish;
101    /** Formulaire permettant l'ajout et l'édition **/
102    private Form form;
103    /** Bas du formulaire **/
104    private HorizontalLayout footerLayout;
105    /** Zone contenant les boutons du formulaire **/
106    private HorizontalLayout buttons;
107    /** Bouton Annuler **/
108    private static Button cancel;
109    /** Informations sur le dossier **/
110    private Folder folder = null;
111    /** Liste des contacts sélectionnés comme destinataires **/
112    private ArrayList<Integer> slctContacts = new ArrayList<Integer>();
113    /** Table qui va contenir les fichiers uploadés **/
114    private Table tblUpload;
115    /** Action supprimé **/
116    private Action ACTION_DEL;
117    /**
118     * Label qui indique de sélectionner les fichiers (sur les navigateur ou le
119     * drag&drop n'est pas implémenté)
120     **/
121    private static Label lblSlctFiles = null;
122    /** Uploader **/
123    private CustomUpload mFileUp;
124    /** Dossier dans lequel nous enverrons nos fichiers **/
125    private String ROOT_DIRECTORY;
126    /** Bouton "Renommer" **/
127    private static Button btRename;
128    /** Indique si nous sommes passé en vue "Renommage" **/
129    private boolean isRenamedView;
130    /** Label qui contient la taille envoyée/restante **/
131    private static Label lblSize;
132    /** Indicateur de progression général **/
133    private static ProgressIndicator superIndicator;
134 
135    /**
```

```java
     * Enregistrement des valeurs récoltées par les Wizard précédent et création
     * du formulaire
     *
     * @param folder
     *            Informations sur le dossier, définies dans le wizard 1
     * @param slctContacts
     *            Destinataires sélectionnés dans le wizard 2
     */
    public WizFoldUpload(Folder folder, ArrayList<Integer> slctContacts) {
        this.folder = folder;
        this.slctContacts = slctContacts;

        ACTION_DEL = new Action(Global.i("CAPTION_DELETE"), new ThemeResource(
                Global.PATH_THEME_RESSOURCES + "false.png"));

        winFolder.setClosable(false);
        winFolder.setWidth("720px");
        winFolder.setHeight("360px");
        winFolder.setCaption(Global.i("CAPTION_TRANSFERT") + " - "
                + Global.i("TITLE_WINDOW_ADD_FILES"));
        winFolder.setIcon(new ThemeResource(Global.PATH_THEME_RESSOURCES
                + "reload.png"));
        winFolder.center();

        doForm();
    }

    /**
     * Construction du formulaire
     */
    private void doForm() {
        mainLayout = new VerticalLayout();
        mainLayout.setSizeFull();
        mainLayout.setSpacing(true);

        // TABLE
        tblUpload = new Table();
        global.setUploadTable(tblUpload);
        tblUpload.setSelectable(true);
        tblUpload.setImmediate(false);
        tblUpload.setMultiSelect(true);
        tblUpload.setColumnReorderingAllowed(true);
        tblUpload.setColumnCollapsingAllowed(true);
        tblUpload.setSizeFull();
        tblUpload.setHeight("215px");
        tblUpload.setVisible(false);

        // tblUpload.addContainerProperty("file_id", String.class, null, "ID",
        // null, null);
        tblUpload.addContainerProperty("file_icon", Embedded.class, null, "",
                null, Table.ALIGN_CENTER);
        tblUpload.addContainerProperty("file_name", String.class, null,
                Global.i("CAPTION_NAME"), null, null);
        tblUpload.addContainerProperty("file_size", String.class, null,
                Global.i("CAPTION_SIZE"), null, null);
        tblUpload.addContainerProperty("file_progress",
                ProgressIndicator.class, null, Global.i("CAPTION_PROGRESS"),
                null, Table.ALIGN_CENTER);
        tblUpload.addContainerProperty("file_etat", String.class, null,
                Global.i("CAPTION_STATUS"), null, Table.ALIGN_CENTER);

        tblUpload.setColumnWidth("file_icon", 24);
```



```java
        tblUpload.setColumnWidth("file_name", 310);
        tblUpload.setColumnWidth("file_size", 55);
        tblUpload.setColumnWidth("file_progress", 150);
        tblUpload.setColumnWidth("file_etat", 70);

        mFileUp = new CustomUpload(folder, slctContacts);

        // Menu contextuel sur les éléments du tableau
        tblUpload.addActionHandler(new Action.Handler() {
            public void handleAction(Action action, Object sender, Object target) {
                if (action == ACTION_DEL) {
                    // Obtention de la liste des fichiers renommés
                    ArrayList<FileUp> fi = mFileUp.getRenamedFileList();

                    Item item = tblUpload.getItem(target);
                    String name = item.getItemProperty("file_name").toString();

                    // Sélection de l'index du tableau ou réside le bon nom de
                    // fichier à supprimer.
                    int indexToDelete = 0;
                    for (int i = 0; i < fi.size(); i++) {
                        if (fi.get(i).getOriginalName().equals(name)) {
                            indexToDelete = i;
                        }
                    }

                    System.out.println("name : " + name + "\n" + "index : "
                            + indexToDelete + "\n" + "nom sur le disque : "
                            + fi.get(indexToDelete).getFileDiskInfo().getName()
                            + "\n");

                    // Suppression physique du fichier renommé
                    File file = new File(Global.UPLOAD_DIR
                            + fi.get(indexToDelete).getFileDiskInfo().getName());
                    if (file.exists()) {
                        mFileUp.decreaseQueueSize(file.length());
                        lblSize.setValue(Utilities.formatSize(mFileUp
                                .getQueueSize())
                                + " / "
                                + Utilities.formatSize(Long.parseLong(Global
                                        .getParm("UPLOAD_MAX_SIZE"))));

                        if (file.delete()) {
                            getWindow().showNotification(null,
                                    Global.i("CAPTION_FILE_DELETED"),
                                    Notification.TYPE_TRAY_NOTIFICATION);
                        }
                    }

                    // Suppression de la ligne du tableau
                    tblUpload.removeItem(target);

                    // Si on a tout supprimé les éléments du tableau
                    if (tblUpload.size() < 1) {
                        btFinish.setEnabled(false);
                        btFinish.setVisible(false);
                        btRename.setVisible(false);
                        cancel.setVisible(true);
                        cancel.setEnabled(true);
                    }
                }
            }
```

```
260
261             public Action[] getActions(Object target, Object sender) {
262                 if (target != null) {
263                     if (mFileUp.getStatus() == 1) {
264                         return new Action[] { ACTION_DEL };
265                     } else {
266                         return null;
267                     }
268                 } else {
269                     return null;
270                 }
271             }
272         });
273         // Clique droit sur les éléments du tableau
274         tblUpload.addListener(new ItemClickListener() {
275             public void itemClick(ItemClickEvent event) {
276                 if (event.getButton() == ItemClickEvent.BUTTON_RIGHT
277                         && mFileUp.getStatus() == 1) {
278                     tblUpload.setValue(null); // Dé-sélectionne tout
279                     tblUpload.select(event.getItemId()); // Sélectionne la ligne
280                                                          // en cours
281                 }
282             }
283         });
284
285         // UPLOAD
286         // -----------------------------------------------------------------
287         mFileUp.setCaption(null);
288
289         ROOT_DIRECTORY = Global.UPLOAD_DIR;
290
291         mFileUp.setRootDirectory(ROOT_DIRECTORY);
292         mFileUp.setUploadButtonCaption(Global.i("CAPTION_UPLOAD_BUTTON"));
293         mFileUp.setAllowedExtentions(Global
294                 .getParm("UPLOAD_ALLOWED_EXTENSIONS"));
295         global.setUploadModule(mFileUp);
296
297         boolean DragAndDropSupported = false;
298         if (Global.browser.isChrome()) {
299             DragAndDropSupported = true;
300         } else if (Global.browser.isFirefox()) {
301             DragAndDropSupported = true;
302         } else if (Global.browser.isSafari()) {
303             DragAndDropSupported = true;
304         }
305
306         // Si le navigateur ne supporte pas le drag&drop
307         if (!DragAndDropSupported) {
308             lblSlctFiles = new Label("<br><br><br><br><br>"
309                     + Global.i("CAPTION_SELECT_FILES"), Label.CONTENT_XHTML);
310             lblSlctFiles.setSizeFull();
311             lblSlctFiles.setHeight("210px");
312             lblSlctFiles.setStyleName("v-label-slct-files");
313             lblSlctFiles.addStyleName("align-center");
314         }
315         // -----------------------------------------------------------------
316
317         mainLayout.addComponent(mFileUp);
318         mainLayout.addComponent(tblUpload);
319         if (lblSlctFiles != null && !DragAndDropSupported) {
320             mainLayout.addComponent(lblSlctFiles);
321         }
```



```
322         global.setUploadModuleLayout(mainLayout);
323
324         buttons = new HorizontalLayout();
325         buttons.setSpacing(true);
326         mainLayout.addComponent(buttons);
327         mainLayout.setComponentAlignment(buttons, Alignment.BOTTOM_RIGHT);
328
329         lblSize = new Label(Utilities.formatSize(0)
330                 + " / "
331                 + Utilities.formatSize(Long.parseLong(Global
332                         .getParm("UPLOAD_MAX_SIZE"))));
333         lblSize.setStyleName("caption-uploadsize");
334         buttons.addComponent(lblSize);
335         buttons.setComponentAlignment(lblSize, Alignment.MIDDLE_LEFT);
336
337         superIndicator = new ProgressIndicator((float) 0);
338         superIndicator.setStyleName("super");
339         superIndicator.setVisible(false);
340         buttons.addComponent(superIndicator);
341         buttons.setComponentAlignment(superIndicator, Alignment.MIDDLE_LEFT);
342
343         cancel = new Button(Global.i("CAPTION_CANCEL"),
344                 new Button.ClickListener() {
345                     public void buttonClick(ClickEvent event) {
346                         (winFolder.getParent()).removeWindow(winFolder);
347                     }
348                 });
349         cancel.setTabIndex(4);
350         buttons.addComponent(cancel);
351         buttons.setComponentAlignment(cancel, Alignment.BOTTOM_RIGHT);
352
353         // Bouton "Renommer"
354         btRename = new Button(Global.i("CAPTION_RENAME_FILES"));
355         btRename.addListener(EditTable);
356         btRename.setVisible(false);
357         buttons.addComponent(btRename);
358         buttons.setComponentAlignment(btRename, Alignment.MIDDLE_RIGHT);
359
360         // Création du bouton "Ajouter"
361         btFinish = new Button(Global.i("CAPTION_FINISH"));
362         btFinish.setStyleName(Runo.BUTTON_DEFAULT);
363         btFinish.setVisible(false);
364         btFinish.setTabIndex(3);
365         btFinish.setClickShortcut(KeyCode.ENTER);
366         btFinish.addListener(SubmitListener);
367         buttons.addComponent(btFinish);
368         buttons.setComponentAlignment(btFinish, Alignment.MIDDLE_RIGHT);
369
370         setCompositionRoot(mainLayout);
371     }
372
373     /**
374      * Retourne le bouton "Terminer"
375      *
376      * @return Bouton "Terminer"
377      */
378     public static Button getBtFinish() {
379         return btFinish;
380     }
381
382     /**
383      * @return Bouton "Renommer"
```

```
384      */
385     public static Button getBtRename() {
386         return btRename;
387     }
388
389     /**
390      * Retourne le bouton "Annuler"
391      *
392      * @return Bouton "Annuler"
393      */
394     public static Button getBtCancel() {
395         return cancel;
396     }
397
398     /**
399      * Retourne le label indiquant de sélectionner les fichier
400      *
401      * @return Label indiquant de sélectionner les fichiers
402      */
403     public static Label getSlctFileLbl() {
404         return lblSlctFiles;
405     }
406
407     /**
408      * @return Label qui contient la taille envoyée/restante
409      */
410     public static Label getLblSize() {
411         return lblSize;
412     }
413
414     /**
415      * @return Indicateur de progression de l'envoi des fichiers
416      */
417     public static ProgressIndicator getSuperIndicator() {
418         return superIndicator;
419     }
420
421     /**
422      * Action sur le clic du bouton "Renommer"
423      */
424     private ClickListener EditTable = new ClickListener() {
425         public void buttonClick(ClickEvent event) {
426             mFileUp.setVisible(false);
427             tblUpload.setStyleName("no-resizable");
428             tblUpload.setHeight("245px");
429             tblUpload.select(null);
430             tblUpload.setValue(null);
431             tblUpload.setSelectable(false);
432             tblUpload.setColumnCollapsingAllowed(false);
433             tblUpload.setStyleName("table-editable");
434             tblUpload.setVisibleColumns(new String[] { "file_name" });
435             tblUpload.addContainerProperty("file_description", String.class,
436                     "", Global.i("CAPTION_DESCRIPTION"), null, null);
437             tblUpload.setColumnWidth("file_name", 240);
438             tblUpload.setEditable(true);
439             tblUpload.setCurrentPageFirstItemIndex(tblUpload
440                     .getCurrentPageFirstItemIndex());
441             tblUpload.setColumnWidth("file_description", 240);
442             tblUpload.removeAllActionHandlers();
443             winFolder.setCaption(Global.i("CAPTION_TRANSFERT") + " - "
444                     + Global.i("CAPTION_RENAME_FILES"));
445             btRename.setVisible(false);
```



```
446             isRenamedView = true;
447         }
448     };
449     /**
450      * Action sur le clic du bouton "Terminer"
451      */
452     private ClickListener SubmitListener = new ClickListener() {
453         public void buttonClick(ClickEvent event) {
454
455             int PKNoFolder = 0;
456
457             // Création du dossier
458             Sql.exec(tbl_folders.getInsertFolder(folder.getName(),
459                     folder.getExpiration(), folder.getDescription()));
460             ResultSet dataF = Sql.query(tbl_folders.getSelectMaxPK());
461             try {
462                 if (dataF.first()) {
463                     PKNoFolder = dataF.getInt("PKNoFolder");
464                 }
465             } catch (SQLException e1) {
466                 e1.printStackTrace();
467             }
468
469             // Enregistrement des destinataires
470             for (int contact : slctContacts) {
471                 Sql.exec(tbl_recipients.getInsertRecipient(PKNoFolder, contact));
472             }
473
474             // Obtention de la liste des fichiers a enregistrer
475             // ------------------------------
476             // Contient tous les fichiers envoyés, même ceux non autorisés
477             ArrayList<FileUp> fi = mFileUp.getRenamedFileList();
478             // Contient tous les fichiers autorisés
479             ArrayList<FileUp> fiWithouthNonAutExt = new ArrayList<FileUp>();
480
481             System.out.println("Liste des fichiers renommés : ");
482
483             // On ajoute à la liste uniquement les fichiers autorisés
484             String extAutorise = Global.getParm("UPLOAD_ALLOWED_EXTENSIONS");
485             for (FileUp f : fi) {
486                 System.out.println(f.getId() + " - " + f.getOriginalName()
487                         + " - " + f.getFileDiskInfo().getName());
488                 if (extAutorise.indexOf(Utilities.getExtension(f
489                         .getFileDiskInfo().getName())) > -1) {
490                     System.out.println("ajout : " + f.getId() + " - "
491                             + f.getOriginalName() + " - "
492                             + f.getFileDiskInfo().getName());
493                     fiWithouthNonAutExt.add(f);
494                 }
495             }
496
497             // Contient les fichiers autorisés et qui sont affichés dans le
498             // tableau (expulsion des fichiers supprimés du tableau)
499             ArrayList<FileUp> listFileOk = new ArrayList<FileUp>(); // Contient
500                                                                     // les
501                                                                     // fichiers
502                                                                     // a
503                                                                     // enregistrer
504             System.out.println("Fichiers sélectionnés : ");
505             for (Object item : tblUpload.getItemIds()) {
506                 int row = Integer.parseInt(item.toString());
507                 listFileOk.add(fiWithouthNonAutExt.get(row - 1));
```

```
508
509                 System.out.println((row - 1)
510                         + " - "
511                         + fi.get(row - 1).getId()
512                         + " - "
513                         + fiWithouthNonAutExt.get(row - 1).getOriginalName()
514                         + " - "
515                         + fiWithouthNonAutExt.get(row - 1).getFileDiskInfo()
516                                 .getName());
517             }
518
519             // Si on est en vue "Renommage", on assigne le nouveau nom donné aux
520             // fichiers
521             if (isRenamedView) {
522                 for (FileUp fileOk : listFileOk) {
523                     Item it = tblUpload.getItem(fileOk.getId());
524                     if (it != null) {
525                         String nomDonneParUtilisateur = it.getItemProperty(
526                                 "file_name").toString();
527                         fileOk.setOriginalName((nomDonneParUtilisateur
528                                 .isEmpty()) ? fileOk.getOriginalName()
529                                 : nomDonneParUtilisateur);
530                         fileOk.setRename(fileOk.getOriginalName());
531                         fileOk.setDescription(it.getItemProperty(
532                                 "file_description").toString());
533                     }
534                 }
535             }
536
537             // Enregistrement du nom des fichiers envoyés
538             String queryInsert = "INSERT INTO trans_files VALUES ";
539             for (FileUp file : listFileOk) {
540                 File fileOnDisk = file.getFileDiskInfo();
541                 // System.out.println(fileOnDisk.getName()+"
    ("+file.getOriginalName()+") -- "+fileOnDisk.length()+" ---
    "+Utilities.getExtension(fileOnDisk.getName()));
542
543                 String nameOnDisk = fileOnDisk.getName();
544                 String originalName = file.getOriginalName().replace(
545                         Utilities.getExtension(file.getOriginalName()), "");
546                 if (originalName.length() >= 50)
547                     originalName = originalName.substring(0, 50);
548                 originalName += Utilities.getExtension(file.getOriginalName());
549
550                 long size = fileOnDisk.length();
551
552                 queryInsert += "(NULL, "
553                         + PKNoFolder
554                         + ", "
555                         + Utilities.formatSQL(nameOnDisk)
556                         + ", "
557                         + Utilities.formatSQL(originalName)
558                         + ","
559                         + size
560                         + ", "
561                         + Utilities.formatSQL(Utilities
562                                 .getExtension(nameOnDisk)) + ", "
563                         + Utilities.formatSQL(file.getDescription()) + "),";
564             }
565             queryInsert = queryInsert.substring(0, queryInsert.length() - 1);
566             Sql.exec(queryInsert);
567
```



```
568             // On récupère les contact sélectionnés comme destinataires dans le
569             // wizard précédent pour leur envoyer un mail
570             ResultSet recipient = Sql.query(tbl_recipients
571                     .getSelectRecipientsFromFolder(PKNoFolder));
572             ResultSet fold = Sql.query(tbl_folders
573                     .getSelectFolderInfos(PKNoFolder));
574
575             try {
576                 int expiration = tbl_folders
577                         .getNumberOfDayFolderIsAvailable(PKNoFolder);
578
579                 String fold_expiration = "", fold_description = "", fold_name = "";
580                 if (fold.first()) {
581                     fold_expiration = fold.getString("folder_expiration");
582                     fold_description = folder.getDescription();
583                     fold_name = fold.getString("folder_name");
584                 }
585
586                 while (recipient.next()) {
587                     String to = recipient.getString("contact_mail");
588                     int PK = recipient.getInt("PKNoContact");
589                     Mailer.sendDownloadLink(to, PKNoFolder, PK,
590                             fold_expiration, expiration, fold_description,
591                             fold_name);
592                 }
593             } catch (SQLException e) {
594                 e.printStackTrace();
595             }
596
597             // Génération de l'archive ZIP ---------------------------------
598             try {
599                 ZipFileWritter zip = new ZipFileWritter(ROOT_DIRECTORY
600                         + Utilities.getFolderArchiveName(folder.getName()));
601
602                 // Remplissage de l'archive
603                 for (FileUp file : listFileOk) {
604                     File fileOnDisk = file.getFileDiskInfo();
605
606                     String name = fileOnDisk.getName();
607                     zip.addFile(ROOT_DIRECTORY + name);
608                 }
609
610                 zip.close();
611             } catch (IOException ex) {
612                 Logger.getLogger(ZipFileWritter.class.getName()).log(
613                         Level.SEVERE, null, ex);
614             }
615
616             // Mise à jour du menu
617             global.getMenuFolderTable().reload();
618
619             // Sélection du dernier dossier de la liste
620             ResultSet data = Sql.query(tbl_folders.getSelectMaxPK());
621             int lastFolder = 0;
622             try {
623                 if (data.first()) {
624                     lastFolder = data.getInt("PKNoFolder");
625                     global.getMenuFolderTable().select(lastFolder);
626                     global.getMenuFolderTable().setValue(lastFolder);
627                     CccvsTransfert.loadModule(new Module(FolderModule
628                             .getBodyRibbonContent(), FolderModule
629                             .getBodyContent(lastFolder)));
```

```
630                 }
631             } catch (SQLException e) {
632                 e.printStackTrace();
633             }
634 
635             Sql.Disconnect();
636 
637             // Fermeture du wizard
638             (winFolder.getParent()).removeWindow(winFolder);
639 
640             CccvsTransfert.mainWindow.showNotification(
641                     Global.i("CAPTION_FOLDER_SENT"),
642                     Global.i("CAPTION_FOLDER_SEND_DESCRIPTION"),
643                     Notification.TYPE_TRAY_NOTIFICATION);
644         }
645     };
646 
647 }
648 
```

PACKAGE

GLOBAL

- Global
- GlobalObjects
- Module
- UserSession

Objets, paramètres, textes, images et configurations stockés pour une utilisation globale.

```java
package global;

import java.io.File;

/**
 * Class de stockage des champs Globaux (qui doivent être accessibles pour
 * toutes les class de l'application).<br>
 * Elle contient les méthodes de récupération des paramètres et des traductions
 * de l'application. Toutes les méthodes et champs de la class sont statiques.
 *
 * @author Dominique Roduit
 */
public class Global {
    // ======================================================================
    // ========================== CONSTANTES ================================
    // ======================================================================
    /** Langue par défaut de l'interface **/
    public static String APP_LANGUAGE = "fr";
    /**
     * URL de l'application. Seulement la partie
     * http://domaine.ch:port/monApplication. Sans les paramètres de l'URL.
     **/
    public static java.net.URL URL = null;
    /**
     * Indique si l'application est exécutée sur un poste de développement ou de
     * production
     **/
    public static boolean isDev = true;
    /**
     * Contient les paramètres du navigateur tel que le nom, la version et la
     * langue
     **/
    public static WebBrowser browser = null;
    /** Contient les paramètres de l'URL s'il y en a **/
    public static String URLParameters = null;
    /**
     * Paramètre très important qui donne le répertoire dans lequel enregistrer
     * les fichier<br>
     * Initialisation dans la class principal {@link CccvsTransfert}
     */
    public static String UPLOAD_DIR = new File(
            System.getProperty("java.io.tmpdir")).getPath()
            + "/";

    // ======================================================================
    // ====================== IMAGES & RESSOURCES ===========================
    // ======================================================================
    /** Répertoire contenant les ressources (images) du thème **/
    public static final String PATH_THEME_RESSOURCES = "../img/";
    /**
     * Répertoire contenant les ressources du thème.<br>
     * A utiliser lorsqu'on écris l'image en HTML<br>
     * ex. :
     */
    public static final String PATH_THEME_RESSOURCES_HTML =
  "/CCCVsTransfert/VAADIN/themes/CccvsDesign/"
            + PATH_THEME_RESSOURCES;
    /** Logo affiché dans le header du layout principal **/
    public static final String IMG_MAIN_LOGO = PATH_THEME_RESSOURCES
            + "logoMainHeader.png";
    /** Image contact **/
    public static final String IMG_CONTACT = PATH_THEME_RESSOURCES
```



```java
            + "contact.png";
    /**
     * Image affichée dans le footer à côté de l'adresse de l'utilisateur
     * connecté
     **/
    public static final String IMG_APPROVED = PATH_THEME_RESSOURCES
            + "tick-shield.png";
    /** Image d'ajout d'un dossier **/
    public static final String IMG_FOLDER_ADD = PATH_THEME_RESSOURCES
            + "newfolder.png";
    /** Image illustrant un téléchargement **/
    public static final String IMG_DOWNLOAD = PATH_THEME_RESSOURCES + "up.png";
    /** Image illustrant une consultation **/
    public static final String IMG_SEARCH = PATH_THEME_RESSOURCES
            + "search.png";
    /** Image illustrant une archive téléchargée **/
    public static final String IMG_ARCHIVE_DOWNLOAD = PATH_THEME_RESSOURCES
            + "archive_down.png";

    public static String IMG_FLAG_COUNTRY = PATH_THEME_RESSOURCES
            + UserSession.getLangue() + ".png";

    // ================== PARAMETRES ================================
    /** HashMap qui contient tous les paramètres **/
    public static final HashMap<String, String> hashParm = new HashMap<String,
  String>();

    /**
     * Retourne un paramètre de l'application enregistré dans la base de
     * données.
     *
     * @param key
     *            La clé pour trouver le paramètre
     * @return (String) Paramètre de la base de données
     */
    public static String getParm(String key) {
        // La première fois, on rempli le HashMap avec tous les paramètres
        if (hashParm.isEmpty()) {
            ResultSet parm = Sql.query("SELECT * FROM trans_app_param");

            try {
                while (parm.next()) {
                    hashParm.put(parm.getString("parm_key"),
                            parm.getString("parm_value"));
                }
            } catch (SQLException e1) {
                e1.printStackTrace();
            }
        }

        // On sélectionne le paramètre
        String param = "";
        if (hashParm.get(key) != null) {
            param = hashParm.get(key);
            param = param.replace("\"", "'");
        }

        return param;
    }

    // ================== TRADUCTIONS ================================
    /** HashMap qui contient toutes les traductions **/
```

```
133     public static final HashMap<String, String> hashTranslate = new HashMap<String,
    String>();
134
135     /**
136      * Fonction "Intertionalize". Elle permet la traduction des textes de
137      * l'application
138      *
139      * @param key
140      *            Mot clé du texte à afficher
141      * @return Le mot traduit dans la langue en cours
142      */
143     public static String i(String key) {
144         // La première fois, on rempli le HashMap avec toutes les traductions
145         if (hashTranslate.isEmpty()) {
146             reloadTranslations();
147         }
148
149         String trans = "";
150         if (hashTranslate.get(key) != null) {
151             trans = hashTranslate.get(key).trim();
152         }
153
154         return trans;
155     }
156
157     /**
158      * Rechargement du tableau contenant les traductions
159      */
160     public static void reloadTranslations() {
161         // On rempli le HashMap avec toutes les traductions
162         hashTranslate.clear();
163         ResultSet translations = Sql.query("SELECT * FROM trans_translate");
164         try {
165             while (translations.next()) {
166                 System.out.println("remplissage : " + APP_LANGUAGE);
167                 hashTranslate.put(
168                         translations.getString("trans_label"),
169                         translations.getString("trans_"
170                                 + APP_LANGUAGE.toLowerCase()));
171             }
172         } catch (SQLException e1) {
173             e1.printStackTrace();
174         }
175     }
176
177 }
178
```



```
1 package global;
2
3 import main.CccvsTransfert;
19
20 /**
21  * Class stockant tous les objets utilisés souvent, qui doivent être accessible
22  * n'importe où dans l'application.<br>
23  * Attention cependant de ne pas utiliser un objet avant qu'il n'ait été créé et
24  * enregistré dans le champ qui lui est approprié. À la différence de la class
25  * Global, les méthodes ne sont pas statiques, la classe doit être instanciée
26  * pour que ses méthodes soient utilisables.
27  *
28  * @author Dominique Roduit
29  *
30  */
31 public class GlobalObjects {
32     // --------- MAIN
33     /**
34      * Fenêtre principale également atteignable avec
35      * {@code CccvsTransfert.mainWindow}
36      **/
37     private Window mainWindow = null;
38     /** Layout principal de l'application **/
39     private MainLayout mainLayout = null;
40     // --------- CONTACT
41     /** Fenêtre pour l'ajout/edition de contact **/
42     private Window windowContact = null;
43     /** Tableau contenant les contacts d'un utilisateur **/
44     private ContactTable tableContact = null;
45     /** Bouton d'ajout des contacts **/
46     private Button btAddContact = null;
47     /** Table des contacts affichée dans le menu **/
48     private MenuContactTable menuContactTable = null;
49     // --------- FOLDER
50     /** Table des dossiers affichée dans le menu **/
51     private MenuFolderTable menuFolderTable = null;
52     /** Bouton d'ajout d'un dossier **/
53     private Button btAddFolder = null;
54     /** Fenêtre contenant les wizard de création d'un dossier **/
55     private Window windowFolder = null;
56     /**
57      * Table contenant les fichiers uploadés (affectée dans
58      * {@link WizFoldUpload}
59      **/
60     private Table uploadTable = null;
61     /** Module d'upload **/
62     private CustomUpload uploadModule = null;
63     /**
64      * Layout qui contient le module d'upload dans la fenêtre de création d'un
65      * dossier
66      **/
67     private VerticalLayout uploadModuleLayout = null;
68     /** Fenêtre pour la redistribution du lien d'accès au dossier **/
69     private Window winReMailLink = null;
70     /** Fenêtre pour la récupération du lien de téléchargement **/
71     private Window winGetLink = null;
72     // --------- ADMIN
73     private MenuAdmin menuAdmin = null;
74
75     /**
76      * Obtention du tableau des fichiers uploadés.
77      *
```

```
78      * @return Tableau des fichiers uploadés.
79      */
80     public Table getUploadTable() {
81         return uploadTable;
82     }
83
84     /**
85      * Stockage de la table d'upload en global
86      *
87      * @param uploadTable
88      *            Table d'upload
89      */
90     public void setUploadTable(Table uploadTable) {
91         this.uploadTable = uploadTable;
92     }
93
94     /**
95      * Obtention de la fenêtre de création d'un dossier
96      *
97      * @return Fenêtre de création d'un dossier
98      */
99     public Window getWindowFolder() {
100        return windowFolder;
101    }
102
103    /**
104     * Stockage de la fenêtre de création d'un dossier
105     *
106     * @param windowFolder
107     *            Fenêtre de création d'un dossier
108     */
109    public void setWindowFolder(Window windowFolder) {
110        this.windowFolder = windowFolder;
111    }
112
113    /**
114     * Obtention du bouton "Nouveau dossier" dossier
115     *
116     * @return Bouton "Nouveau dossier"
117     */
118    public Button getBtAddFolder() {
119        return btAddFolder;
120    }
121
122    /**
123     * Stockage en global du bouton "Nouveau dossier".
124     *
125     * @param btAddFolder
126     *            Bouton "Nouveau dossier"
127     */
128    public void setBtAddFolder(Button btAddFolder) {
129        this.btAddFolder = btAddFolder;
130    }
131
132    /**
133     * Obtention du tableau des dossiers intégré au menu
134     *
135     * @return Tableau des dossiers du menu
136     */
137    public MenuFolderTable getMenuFolderTable() {
138        return menuFolderTable;
139    }
```



```
140
141    /**
142     * Stockage en global du tableau des dossiers intégré au menu
143     *
144     * @param menuFolderTable
145     *            Tableau des dossiers du menu
146     */
147    public void setMenuFolderTable(MenuFolderTable menuFolderTable) {
148        this.menuFolderTable = menuFolderTable;
149    }
150
151    /**
152     * Obtention du tableau des contacts intégré au menu
153     * {@link MenuContactTable}
154     *
155     * @return Tableau des contacts dumenu
156     */
157    public MenuContactTable getMenuContactTable() {
158        return menuContactTable;
159    }
160
161    /**
162     * Stockage en global du tableau des contacts intégré au menu.
163     * {@link MenuContactTable}
164     *
165     * @param menuContactTable
166     *            Tableau des contacts du menu
167     */
168    public void setMenuContactTable(MenuContactTable menuContactTable) {
169        this.menuContactTable = menuContactTable;
170    }
171
172    /**
173     * Obtention du bouton "Ajouter un contact" {@link ContactModule}
174     *
175     * @return Bouton "Ajouter un contact"
176     */
177    public Button getBtAddContact() {
178        return btAddContact;
179    }
180
181    /**
182     * Stockage en global du bouton "Ajouter un contact". {@link ContactModule}
183     *
184     * @param btAddContact
185     *            Bouton "Ajouter un contact"
186     */
187    public void setBtAddContact(Button btAddContact) {
188        this.btAddContact = btAddContact;
189    }
190
191    /**
192     * Obtention du tableau des contacts. {@link ContactTable}
193     *
194     * @return Tableau des contacts
195     */
196    public ContactTable getTableContact() {
197        return tableContact;
198    }
199
200    /**
201     * Stockage en global du tableau des contacts. {@link ContactTable}
```

```
202      *
203      * @param tableContact
204      *            Tableau des contacts
205      */
206     public void setTableContact(ContactTable tableContact) {
207         this.tableContact = tableContact;
208     }
209
210     /**
211      * Obtention de la fenêtre contenant le formulaire d'ajout/édition de
212      * contact. {@link ContactModule}
213      *
214      * @return Fenêtre d'ajout/édition de contact
215      */
216     public Window getWindowContact() {
217         return windowContact;
218     }
219
220     /**
221      * Stockage en global de la fenêtre d'ajout/édition de contact
222      *
223      * @param windowContact
224      *            Fenêtre d'ajout/édition de contact
225      */
226     public void setWindowContact(Window windowContact) {
227         this.windowContact = windowContact;
228     }
229
230     /**
231      * Obtention du layout principal de l'application. {@link MainLayout}
232      *
233      * @return Layout principal de l'application
234      */
235     public MainLayout getMainLayout() {
236         return mainLayout;
237     }
238
239     /**
240      * Stockage en global du layout principal de l'application. Définit dans
241      * {@link CccvsTransfert}
242      *
243      * @param mainLayout
244      *            Layout principal de l'application
245      */
246     public void setMainLayout(MainLayout mainLayout) {
247         this.mainLayout = mainLayout;
248     }
249
250     /**
251      * @return the uploadModule
252      */
253     public CustomUpload getUploadModule() {
254         return uploadModule;
255     }
256
257     /**
258      * @param uploadModule
259      *            the uploadModule to set
260      */
261     public void setUploadModule(CustomUpload uploadModule) {
262         this.uploadModule = uploadModule;
263     }
```



```
264
265     /**
266      * @return the uploadModuleLayout
267      */
268     public VerticalLayout getUploadModuleLayout() {
269         return uploadModuleLayout;
270     }
271
272     /**
273      * @param uploadModuleLayout
274      *            the uploadModuleLayout to set
275      */
276     public void setUploadModuleLayout(VerticalLayout uploadModuleLayout) {
277         this.uploadModuleLayout = uploadModuleLayout;
278     }
279
280     /**
281      * @return Fenêtre de reditribution du lien d'accès au dossier
282      */
283     public Window getWinReMailLink() {
284         return winReMailLink;
285     }
286
287     /**
288      * @param winReMailLink
289      *            Fenêtre de reditribution du lien d'accès au dossier
290      */
291     public void setWinReMailLink(Window winReMailLink) {
292         this.winReMailLink = winReMailLink;
293     }
294
295     /**
296      * @return Fenêtre pour la récupération du lien de téléchargement
297      */
298     public Window getWinGetLink() {
299         return winGetLink;
300     }
301
302     /**
303      * @param winGetLink
304      *            Fenêtre pour la récupération du lien de téléchargement
305      */
306     public void setWinGetLink(Window winGetLink) {
307         this.winGetLink = winGetLink;
308     }
309
310     /**
311      * Ouvre la fenêtre de récupération du lien de téléchargement
312      *
313      * @param filename
314      *            Nom du fichier pour lequel on veux obtenir le lien
315      */
316     public void openWindowGetLink(String filename) {
317         Window winGetLink = new Window(Global.i("TITLE_WINDOW_GET_LINK"));
318         winGetLink.setIcon(new ThemeResource(Global.PATH_THEME_RESSOURCES
319                 + "link.png"));
320         winGetLink.setResizable(false);
321         winGetLink.setModal(true);
322         winGetLink.addComponent(new GetLinkModule(filename));
323         winGetLink.setWidth("700px");
324         winGetLink.setHeight("176px");
325         CccvsTransfert.getGlobalMethod().setWinGetLink(winGetLink);
```

```
326         CccvsTransfert.mainWindow.addWindow(winGetLink);
327     }
328
329     /**
330      * @return Le menu d'administration
331      */
332     public MenuAdmin getMenuAdmin() {
333         return menuAdmin;
334     }
335
336     /**
337      * @param menuAdmin
338      *            Menu d'administration
339      */
340     public void setMenuAdmin(MenuAdmin menuAdmin) {
341         this.menuAdmin = menuAdmin;
342     }
343 }
344
```



```
1 package global;
2
3 import modules.ContactModule;
8
9 /**
10  * Création d'un module dans le but de l'afficher sur la page.<br>
11  * Un module se compose de deux parties :<br>
12  * <li>1. Contenu du ruban = Zone haute du corps de page qui contient les
13  * boutons et les titres</li> <li>2. Contenu du corps de page = Contenu
14  * principal affiché dans la partie principale de l'application</li> <br>
15  * <br>
16  * Exemple d'appel :
17  * {@code Module MODULE_TRANSFERT = new
    Module(TransfertModule.getBodyRibbonContent(), TransfertModule.getBodyContent()); }
18  *
19  * @author Dominique Roduit
20  *
21  */
22 public class Module extends CustomComponent {
23     /** Contenu du ruban (Boutons, titre, ...) **/
24     private Component ribbonContent;
25     /** Contenu du corps de page **/
26     private Component bodyContent;
27
28     /**
29      * Création d'une instance d'un module
30      *
31      * @param ribbonContent
32      *            Contenu du ruban
33      * @param bodyContent
34      *            Contenu du corps de page
35      */
36     public Module(Component ribbonContent, Component bodyContent) {
37         setRibbonContent(ribbonContent);
38         setBodyContent(bodyContent);
39     }
40
41     /**
42      * Obtention du contenu du corps de page
43      *
44      * @return Contenu du corps de page
45      */
46     public Component getBodyContent() {
47         return bodyContent;
48     }
49
50     /**
51      * Enregistre le contenu du corps de page
52      *
53      * @param bodyContent
54      *            Contenu du corps de page
55      */
56     public void setBodyContent(Component bodyContent) {
57         this.bodyContent = bodyContent;
58     }
59
60     /**
61      * Obtention du contenu du ruban
62      *
63      * @return Contenu du ruban
64      */
65     public Component getRibbonContent() {
```

```
66          return ribbonContent;
67      }
68
69      /**
70       * Enregistrement du contenu du ruban
71       *
72       * @param ribbonContent
73       *            Contenu du ruban
74       */
75      public void setRibbonContent(Component ribbonContent) {
76          this.ribbonContent = ribbonContent;
77      }
78  }
79
```



```
1 package global;
2
3 /**
4  * Stockage des informations d'un utilisateur lors de sa connexion.<br>
5  * Cette class est comparable à une variable de session en PHP.<br>
6  * Lors du rechargement de la page, les données sont conservées grâce à un
7  * ThreadLocal.<br>
8  * Si l'application est redémarrée, la session est détruite.
9  *
10 * @author Dominique Roduit
11 *
12 */
13 public class UserSession {
14     /** Clé primaire de l'utilisateur **/
15     private static int ID = 0;
16     /** Adresse e-mail de l'utilisateur **/
17     private static String mail = "";
18     /** Indique si l'utilisateur est un interne ou non **/
19     private static boolean internal = false;
20     /** Stocke la langue préférée de l'utilisateur **/
21     private static String langue = "fr";
22     /** Indique si l'utilisateur est un administrateur ou non **/
23     private static boolean admin = false;
24
25     /**
26      * Retourne la clé primaire de l'utilisateur
27      *
28      * @return Clé primaire de l'utilisateur
29      */
30     public static int getID() {
31         return ID;
32     }
33
34     /**
35      * Enregistre la clé primaire de l'utilisateur
36      *
37      * @param iD
38      *            Clé primaire de l'utilisateur
39      */
40     public static void setID(int iD) {
41         ID = iD;
42     }
43
44     /**
45      * Retourne l'adresse e-mail de l'utilisateur connecté
46      *
47      * @return Adresse e-mail
48      */
49     public static String getMail() {
50         return mail;
51     }
52
53     /**
54      * Enregistre l'adresse e-mail de l'utilisateur connecté
55      *
56      * @param mail
57      *            Adresse e-mail
58      */
59     public static void setMail(String mail) {
60         UserSession.mail = mail;
61     }
62
```

```java
    /**
     * Indique si l'utilisateur est un interne ou non
     *
     * @return true si l'utilisateur est un interne
     */
    public static boolean isInternal() {
        return internal;
    }

    /**
     * Enregistre si l'utilisateur est un interne ou non
     *
     * @param internal
     *            true si l'utilisateur est un interne
     */
    public static void setInternal(boolean internal) {
        UserSession.internal = internal;
    }

    /**
     * Retourne la langue définit par l'utilisateur ou par son navigateur si
     * celui-ci ne l'a jamais modifiée
     *
     * @return Langue de l'utilisateur
     */
    public static String getLangue() {
        return langue.substring(0, 2);
    }

    /**
     * Enregistre la langue choisie par l'utilisateur
     *
     * @param langue
     *            Langue choisie par l'utilisateur
     */
    public static void setLangue(String langue) {
        UserSession.langue = langue;
        Global.APP_LANGUAGE = langue;
        Global.reloadTranslations();
        Global.IMG_FLAG_COUNTRY = Global.PATH_THEME_RESSOURCES
                + Global.APP_LANGUAGE + ".png";
        System.out.println("Modification de la langue de l'interface : "
                + langue.toUpperCase() + "; Utilisateur : " + ID);
    }

    /**
     * @return Indique si l'utilisateur est un administrateur
     */
    public static boolean isAdmin() {
        return admin;
    }

    /**
     * @param Définit
     *            l'utilisateur comme administrateur ou non
     */
    public static void setAdmin(boolean admin) {
        UserSession.admin = admin;
    }
}
```

PACKAGE

MAIN

- CccvsTransfert
- ConnexionLayout
- MainLayout

Classe principale de l'application (index) et layouts de l'interface utilisateur.

```java
package main;

import global.Global;

/**
 * <b>Application CCCVs-Transfert</b><br>
 * Class principale de l'application qui instancie les composants racines
 * (Fenêtres et Layouts de base).<br>
 * C'est cette classe qui est appelée au lancement de l'application.<br>
 * Elle implémente un ThreadLocal pour conserver l'état de l'application lors
 * d'un rechargement de la page.<br>
 * Si l'application est redémarrée (?restartApplication), le ThreadLocal est
 * réinitialisé.
 *
 * @author Roduit Dominique
 * @version 1.0
 * @since 2013-02-28
 */
public class CccvsTransfert extends Application implements
        HttpServletRequestListener {
    /**
     * ThreadLocal qui stocke l'état de l'application pour le restituer lors du
     * rechargement de la page
     **/
    private static ThreadLocal<CccvsTransfert> threadLocal = new
ThreadLocal<CccvsTransfert>();
    /** Contient tous les objets enregistrés en global **/
    private static GlobalObjects global = new GlobalObjects();
    /**
     * Fenêtre principale qui doit être accessible depuis n'importe ou. Pas de
     * getter/setter parce qu'on ne peut pas le faire pour la fenêtre racine
     **/
    public static Window mainWindow;
    /** Analyse l'URL et réagit en fonction des ancres reçues **/
    private final UriFragmentUtility uri = new UriFragmentUtility();
    /** Composant personnalisé contenant le layout de l'interface utilisateur **/
    private static MainLayout mainLayout;
    /** Composant personnalisé contenant le layout de connexion à l'application **/
    private ConnexionLayout connexionLayout;
    /** Fenêtre flottante intégrée dans le layout de connexion **/
    private Window winConnexion;

    /**
     * Méthode appelée à l'initialisation de l'application.<br>
     * Cette méthode est rappelée chaque fois que le paramètre URL
     * ?restartApplication existe.
     */
    @Override
    public void init() {
        Global.URL = getURL();
        Global.isDev = (Global.URL.toString().indexOf("localhost") > -1 ||
Global.URL
                .toString().indexOf("127.0.0.1") > -1) ? true : false;
        Global.UPLOAD_DIR = (Global.isDev) ? "/wamp/www/files/"
                : Global.UPLOAD_DIR;

        // Application d'un thème personnalisé basé sur un thème natif de vaadin
        setTheme("CccvsDesign");

        // Définition de la fenêtre principal
        mainWindow = new Window(Global.getParm("APP_NAME"));
        mainWindow.setSizeFull();
```



```java
        mainWindow.getContent().setSizeFull();
        setMainWindow(mainWindow);
        // global.setMainWindow(mainWindow);

        // Enregistre dans browser des informations sur le navigateur de
        // l'utilisateur
        WebApplicationContext context = ((WebApplicationContext) mainWindow
                .getApplication().getContext());
        Global.browser = context.getBrowser();

        // Suppression des marges de 18px du body générées automatiquement par
        // vaadin
        AbstractLayout panelLayout = (AbstractLayout) mainWindow.getContent();
        panelLayout.setMargin(false);

        // Création du Layout de connexion
        createConnexionLayout();

        // Action executées lors d'interception de paramètres dans l'URL
        setNavigationByURLParameters();
        // setNavigationByURLAnchors();
    }

    // ------------------------------------------------------------------------
---
    /**
     * Commence l'implémentation du ThreadLocal
     *
     * @return Application enregistrée dans le threadLocal
     */
    public static CccvsTransfert getInstance() {
        return threadLocal.get();
    }

    /**
     * Remplace l'état en cours de l'application par celui enregistrée dans le
     * Thread
     *
     * @param application
     *            L'application à remplacer
     */
    public static void setInstance(CccvsTransfert application) {
        threadLocal.set(application);
    }

    /**
     * Méthode appelée avant le chargement de l'application, elle permet de
     * rétablir l'état de l'application avant le rechargement de la page.
     */
    public void onRequestStart(HttpServletRequest request,
            HttpServletResponse response) {
        CccvsTransfert.setInstance(this);
    }

    /**
     * Méthode appelée à la fin de chaque requête
     */
    public void onRequestEnd(HttpServletRequest request,
            HttpServletResponse response) {
        threadLocal.remove();
    }
```

```
    // ------------------------------------------------------------------------
    ---
    /**
     * Méthode très importante qui renvoie les objets stockés en global
     *
     * @return (GlobalMethod) Instance stockées dans la class GlobalMethod
     */
    public static GlobalObjects getGlobalMethod() {
        return global;
    }

    /**
     * Configuration des différents messages du système VAADIN, par exemple,
     * lorsque les cookies sont désactivés ou que la session a expirée.
     *
     * @return Messages personnalisés
     */
    public static SystemMessages getSystemMessages() {
        CustomizedSystemMessages messages = new CustomizedSystemMessages();
        messages.setSessionExpiredCaption(Global
                .i("MESS_SYS_SESSION_EXPIRED_CAPTION"));
        messages.setSessionExpiredMessage(Global
                .i("MESS_SYS_SESSION_EXPIRED_MESSAGE"));
        messages.setCookiesDisabledCaption(Global
                .i("MESS_SYS_COOKIES_DISABLED_CAPTION"));
        messages.setCookiesDisabledMessage(Global
                .i("MESS_SYS_COOKIES_DISABLED_MESSAGE"));
        messages.setAuthenticationErrorCaption(Global
                .i("MESS_SYS_AUTH_ERROR_CAPTION"));
        messages.setAuthenticationErrorMessage(messages
                .getSessionExpiredMessage());
        messages.setCommunicationErrorCaption(Global
                .i("MESS_SYS_COMM_ERROR_CAPTION"));
        messages.setCommunicationErrorMessage(messages
                .getSessionExpiredMessage());
        messages.setInternalErrorCaption(Global
                .i("MESS_SYS_INTERNAL_ERROR_CAPTION"));
        messages.setInternalErrorMessage(messages.getSessionExpiredMessage());
        return messages;
    }

    /**
     * Reste à l'écoute des paramètres de l'URL pour définir les pages selon les
     * paramètres
     */
    private void setNavigationByURLParameters() {
        // On reste à l'écoute des éventuels paramètres de l'URL qui pourraient
        // survenir
        ParameterHandler URLParameter = new ParameterHandler() {
            @Override
            public void handleParameters(Map<String, String[]> parameters) {
                // Récupération des paramètres et placement dans un tableau
                ArrayList<String> keys = new ArrayList<String>();
                HashMap<String, String> parm = new HashMap<String, String>();
                int index = 0;
                for (Iterator it = parameters.keySet().iterator(); it.hasNext();) {
                    keys.add((String) it.next());
                    parm.put(keys.get(index),
                            ((String[]) parameters.get(keys.get(index)))[0]);

                    Global.URLParameters += keys.get(index) + "="
                            + parm.get(keys.get(index)) + "&";
```



```
                    index++;
                }

                // On parcours les paramètres
                for (Iterator it = parameters.keySet().iterator(); it.hasNext();) {
                    String key = (String) it.next();
                    String value = ((String[]) parameters.get(key))[0];

                    // Si on tente d'accéder à la page de téléchargement
                    if (key.equals(Global.getParm("URL_PARM_DOWNLOAD"))) {

                        // Obtention des informations sur le contact qui accède
                        // à la page
                        ResultSet contact = Sql.query(tbl_contacts
                                .getContactByCryptedURL(parm.get("id")));
                        int PKNoContact = 0;
                        String mail = "";
                        try {
                            if (contact.first()) {
                                mail = contact.getString("contact_mail");
                                if (SHA256.getHashValue(
                                        Global.getParm("SECRET_KEY") + mail)
                                        .equals(parm.get("idm"))) {
                                    PKNoContact = contact.getInt("PKNoContact");

                                    UserSession.setMail(mail);
                                    UserSession.setLangue(Global.browser
                                            .getLocale().toString()
                                            .substring(0, 2));
                                }
                            }
                        } catch (SQLException e1) {
                            e1.printStackTrace();
                        }

                        // Obtention des informations sur le dossier demandé
                        ResultSet folder = Sql.query(tbl_folders
                                .getFolderByCryptedURL(value));
                        int PKNoFolder = 0;
                        try {
                            if (folder.first()) {
                                if (!folder.getBoolean("folder_archive")) {
                                    if (PKNoContact > 0) {
                                        createMainLayout(false);
                                        mainLayout
                                                .setMenuWidth(Global
                                                        .getParm("LAYOUT_FOLDER_INF
O_WIDTH"));
                                        mainLayout.getBodyRibbon()
                                                .setSizeFull();
                                        mainLayout.setFooterEnabled(true);
                                        mainLayout.getMenuRibbon()
                                                .addStyleName("expand");

                                        PKNoFolder = folder
                                                .getInt("PKNoFolder");
                                        DownloadModule downMod = new
DownloadModule(
                                                PKNoFolder, PKNoContact, mail);
                                        loadModule(new Module(
                                                downMod.getBodyRibbonContent(),
                                                downMod.getBodyContent()));
```

```
283                                     } else {
284                                         mainWindow
285                                                 .showNotification(
286                                                         Global.i("CAPTION_ERROR"),
287                                                         Global.i("CAPTION_USER_NOT_A
    LLOWED"),
288                                                         Notification.TYPE_ERROR_MES
    SAGE);
289                                     }
290                                 } else {
291                                     mainWindow.showNotification(Global
292                                             .i("CAPTION_EXPIRATION"), Global
293                                             .i("CAPTION_FOLDER_NOT_AVAILABLE"),
294                                             Notification.TYPE_ERROR_MESSAGE);
295                                 }
296                             } else {
297                                 mainWindow.showNotification(
298                                         Global.i("CAPTION_ERROR"),
299                                         Global.i("CAPTION_FOLDER_NOT_EXISTS"),
300                                         Notification.TYPE_ERROR_MESSAGE);
301                             }
302                         } catch (SQLException e) {
303                             e.printStackTrace();
304                         }
305
306                         // getMainWindow().showNotification(value);
307                     }
308
309                     // On reçoit une demande de validation
310                     if (key.equals(Global.getParm("URL_PARM_VALIDATION"))) {
311                         // Sélection de l'utilisateur en se basant sur la valeur
312                         // du paramètre crypté passé dans l'URL
313                         ResultSet user = Sql.query(tbl_users
314                                 .getUserByCryptedURL(value));
315                         try {
316                             if (user.first()) {
317                                 if (!user.getBoolean("user_validation")) {
318                                     getMainWindow()
319                                             .showNotification(
320                                                     Global.i("CAPTION_MAIL_VALIDATI
    ON_OK"),
321                                                     user.getString("user_mail"),
322                                                     Notification.TYPE_TRAY_NOTIFICA
    TION);
323
324                                     // Validation de l'utilisateur
325                                     Sql.exec(tbl_users
326                                             .getUpdByMailValidation(user
327                                                     .getString("user_mail")));
328                                     // Journalisation de la validation
329                                     Sql.exec(tbl_journal_con.getInsertQuery(
330                                             user.getString("user_mail"),
331                                             "validation_success",
332                                             "Validation réussie"));
333
334                                     // Enregistrement des informations sur
335                                     // l'utilisateur
336                                     UserSession.setID(user.getInt("PKNoUser"));
337                                     UserSession.setMail(user
338                                             .getString("user_mail"));
339                                     UserSession.setInternal(user
340                                             .getBoolean("user_internal"));
```



```
341                                     UserSession.setLangue(user
342                                             .getString("user_langue"));
343                                     UserSession.setAdmin(user
344                                             .getBoolean("user_admin"));
345
346                                     // Connexion automatique
347                                     createMainLayout(true);
348                                 } else {
349                                     getMainWindow()
350                                             .showNotification(
351                                                     Global.i("CAPTION_MAIL_ALREADY_V
    ALIDATE"),
352                                                     user.getString("user_mail"),
353                                                     Notification.TYPE_WARNING_MESSA
    GE);
354
355                                     // Journalisation de la validation déjà
356                                     // effectuée
357                                     Sql.exec(tbl_journal_con.getInsertQuery(
358                                             user.getString("user_mail"),
359                                             "validation_recurrent",
360                                             "Relancement du lien alors que
    l'adresse est déjà validée"));
361                                 }
362                             } else {
363                                 // Journalisation de l'erreur
364                                 Sql.exec(tbl_journal_con
365                                         .getInsertQuery(value,
366                                                 "validation_error",
367                                                 "Essai de validation d'une adresse
    qui n'existe pas !"));
368                             }
369                         } catch (SQLException e) {
370                             e.printStackTrace();
371                         }
372                     }
373                 }
374             }
375         };
376         mainWindow.addParameterHandler(URLParameter);
377     }
378
379     /**
380      * Reste à l'écoute des ancres de l'URL pour définir les pages selon l'ancre
381      * reçue
382      */
383     private void setNavigationByURLAnchors() {
384         uri.addListener(new UriFragmentUtility.FragmentChangedListener() {
385             private static final long serialVersionUID = -3992570785776598885L;
386
387             @Override
388             public void fragmentChanged(FragmentChangedEvent source) {
389                 getMainWindow().showNotification(
390                         source.getUriFragmentUtility().getFragment());
391             }
392
393         });
394         mainWindow.addComponent(uri);
395     }
396
397     /**
398      * Création du layout principal
```

```
399      *
400      * @param loadMainComponents
401      *            true : Charger les composants, false : ne charger que le
402      *            layout de base
403      */
404     private void createMainLayout(boolean loadMainComponents) {
405         System.out.println("Chargement du layout principal");
406
407         // Désactive le layout de connexion
408         if (connexionLayout != null) {
409             mainWindow.removeComponent(connexionLayout);
410             mainWindow.removeWindow(winConnexion);
411         }
412
413         // Désactive le layout principal s'il existe déjà
414         if (mainLayout != null) {
415             mainWindow.removeComponent(mainLayout);
416         }
417
418         // Ajoute le design personnalisé à la page
419         mainLayout = new MainLayout();
420         mainLayout.setFooterEnabled(false);
421         mainWindow.addComponent(mainLayout);
422         global.setMainLayout(mainLayout);
423
424         // Ajout du logo
425         // ------------------------------------------------------------
426         Embedded logo = new Embedded();
427         logo.setSource(new ThemeResource(Global.IMG_MAIN_LOGO));
428         mainLayout.getHeaderLogo().addComponent(logo);
429
430         // Ajout des composants
431         if (loadMainComponents)
432             createMainComponents();
433         else
434             mainLayout.getBtChangeLanguage().setVisible(false);
435     }
436
437     /**
438      * Chargement des composants du layout principal
439      */
440     private void createMainComponents() {
441         // Ajout de bouton dans le menu du header
442         // ------------------------------
443         Button btAbout = new Button(Global.i("CAPTION_ABOUT"),
444                 new Button.ClickListener() {
445                     public void buttonClick(ClickEvent event) {
446                         loadModule(new Module(
447                                 AboutModule.getBodyRibbonContent(),
448                                 AboutModule.getBodyContent()));
449                     }
450                 });
451         Button btFileManager = new Button(Global.i("CAPTION_USAGE_CONDITIONS"),
452                 new Button.ClickListener() {
453                     public void buttonClick(ClickEvent event) {
454                         loadModule(new Module(
455                                 ConditionsModule.getBodyRibbonContent(),
456                                 ConditionsModule.getBodyContent()));
457                     }
458                 });
459         mainLayout.getHeaderMenu().addComponent(btFileManager);
460         mainLayout.getHeaderMenu().addComponent(btAbout);
```



```
461
462         // Bouton de changement de langue
463         mainLayout.getBtChangeLanguage().addListener(
464                 new Button.ClickListener() {
465                     public void buttonClick(ClickEvent event) {
466                         // Suppression du layout affiché
467                         mainWindow.removeComponent(mainLayout);
468                         // Rechargement complet du layout
469                         createMainLayout(true);
470                         // Affichage d'une notification
471                         String langue = (UserSession.getLangue().equals("fr")) ?
    Global
472                                 .i("CAPTION_FRENCH") : Global
473                                 .i("CAPTION_GERMAN");
474                         mainWindow.showNotification(
475                                 Global.i("CAPTION_LANGUE_CHANGE"), langue,
476                                 Notification.TYPE_TRAY_NOTIFICATION);
477                     }
478                 });
479
480         // Ajout du contenu du menu ----------------------------------------
481         AccordionMenu menu = new AccordionMenu();
482         menu.setSizeFull();
483         mainLayout.getMenu().addComponent(menu);
484         // -----------------------------------------------------------------
485
486         // Chargement des modules (= pages) ---------------------------------
487         if (tbl_contacts.getNumberOfContact() < 1) {
488             loadModule(new Module(ContactModule.getBodyRibbonContent(),
489                     TransfertModule.getBodyContent()));
490         } else {
491             loadModule(new Module(TransfertModule.getBodyRibbonContent(),
492                     TransfertModule.getBodyContent()));
493         }
494         // ------------------------------------------------------------------
495     }
496
497     /**
498      * Charge un module dans l'application
499      *
500      * @param module
501      *            (Module) Le module à charger
502      */
503     public static void loadModule(Module module) {
504         if (mainLayout != null) {
505             mainLayout.restoreRibbonHeight();
506
507             if (module.getRibbonContent() != null) {
508                 mainLayout.getBodyRibbon().removeAllComponents();
509                 mainLayout.getBodyRibbon().addComponent(
510                         module.getRibbonContent());
511             }
512             if (module.getBodyContent() != null) {
513                 mainLayout.getBody().removeAllComponents();
514                 mainLayout.getBody().addComponent(module.getBodyContent());
515             }
516         }
517     }
518
519     /**
520      * Supprime le contenu de la page en cours
521      */
```

```
522     public static void resetContent() {
523         if (mainLayout != null) {
524             mainLayout.getBody().removeAllComponents();
525             mainLayout.getBodyRibbon().removeAllComponents();
526         }
527     }
528 
529     /**
530      * Création du layout de connexion
531      */
532     private void createConnexionLayout() {
533         System.out.println("Chargement du layout de connexion");
534 
535         // Désactive le layout principal
536         if (mainLayout != null) {
537             mainWindow.removeComponent(mainLayout);
538         }
539 
540         connexionLayout = new ConnexionLayout();
541         mainWindow.addComponent(connexionLayout);
542 
543         winConnexion = connexionLayout.getWindowConnexion();
544         winConnexion.center();
545         mainWindow.addWindow(winConnexion);
546 
547         connexionLayout.getTextFieldID().focus();
548 
549         // On écoute quand le formulaire est validé
550         connexionLayout.getButtonConnexion().addListener(
551                 new Button.ClickListener() {
552                     public void buttonClick(ClickEvent event) {
553                         if (connexionLayout.getConnexionState()) {
554                             createMainLayout(true);
555                         }
556                     }
557                 });
558     }
559 
560 }
561 
```



```
1 package main;
2 
3 import global.Global;
43 
44 /**
45  * <b>IMPORTANT !<b><br>
46  * <br>
47  * Construction du composant de connexion à l'application.<br>
48  * <br>
49  * Ce composant personnalisé est un layout qui contient le composant de
50  * connexion à l'interface utilisateur. La class est instanciée depuis
51  * {@link CccvsTransfert}.
52  *
53  * @author Dominique Roduit
54  * @version 1.0.0
55  * @since 2013-03-01
56  *
57  */
58 public class ConnexionLayout extends CustomComponent {
59     /** Hauteur du header **/
60     private final String HEADER_HEIGHT = Global
61             .getParm("LAYOUT_CONNEXION_HEADER_HEIGHT");
62     /**
63      * Layout principal pleine page qui contient les trois niveaux de bases
64      * (top, middle, bottom)
65      **/
66     private AbsoluteLayout mainLayout;
67 
68     // Zone du haut
69     // --------------------------------------------------------------------
70     /** Layout général du haut **/
71     private HorizontalLayout topLayout;
72 
73     // Zone du milieu qui contient le corps de page -------------------------
74     /** Layout général du milieu **/
75     private HorizontalLayout middleLayout;
76     /** Logo de la fenêtre flottante **/
77     private VerticalLayout logo;
78     /** Layout contenant le formulaire de connexion **/
79     private VerticalLayout velConnexion;
80     /**
81      * Bouton de connexion/inscription.<br>
82      * Selon le statut de l'utilisateur (enregistré ou non) le bouton change
83      **/
84     private Button btnConn;
85     /** Champs pour le mot de passe **/
86     private PasswordField txtPass;
87     /** Champs pour l'identifiant (e-mail) **/
88     private TextField txtIdentifiant;
89     /** Layout conteneur du logo **/
90     private VerticalLayout containerLogo;
91 
92     /** Fenêtre flottante de connexion **/
93     private Window winConnexion;
94 
95     /** Stocke si l'utilisateur existe déjà dans la base de données ou non **/
96     private boolean existingUser = false;
97     /** Stocke l'état de connexion (Connexion ok ou non) **/
98     private Boolean connexionState = false;
99 
100     /**
101      * Construction des 3 niveaux de bases (top, middle, bottom)
```

```
102     */
103    public ConnexionLayout() {
104        super.setSizeFull();
105
106        // Absolute Layout qui va contenir tout les autres layout
107        mainLayout = new AbsoluteLayout();
108        mainLayout.setSizeFull();
109        mainLayout.setStyleName("connexion");
110
111        // Grid pleine page qui permet de placer les 3 premiers niveau de Layout
112        // Verticalement
113        GridLayout vMain = new GridLayout(1, 3);
114        vMain.setSizeFull();
115        vMain.setRowExpandRatio(1, 100);
116        mainLayout.addComponent(vMain);
117
118        // Header
119        topLayout = new HorizontalLayout();
120        topLayout.setHeight(HEADER_HEIGHT);
121        topLayout.setWidth("100%");
122        topLayout.setStyleName("v-header");
123        vMain.addComponent(topLayout, 0, 0);
124
125        // Body
126        middleLayout = new HorizontalLayout();
127        middleLayout.setSizeFull();
128        middleLayout.setStyleName("v-body");
129        vMain.addComponent(middleLayout, 0, 1);
130
131        setCompositionRoot(mainLayout);
132    }
133
134    /**
135     * Retourne la fenêtre flottante contenant le contenu
136     *
137     * @return Fenêtre flottante
138     */
139    public Window getWindowConnexion() {
140        // Création de la fenetre flottante
141        winConnexion = new Window(Global.getParm("APP_NAME") + " "
142                + Global.getParm("APP_VERSION"));
143
144        // connexionWindow.setModal(true);
145        winConnexion.setStyleName(Reindeer.LAYOUT_BLACK);
146
147        // Taille de la fenêtre de connexion
148        winConnexion.setHeight("415px");
149        winConnexion.setWidth("420px");
150
151        // Positionnement de la fenetre flottante
152        winConnexion.center();
153
154        // Défense de fermer et de redimensionner la fenêtre flottante
155        winConnexion.setClosable(false);
156        winConnexion.setResizable(false);
157
158        containerLogo = new VerticalLayout();
159        containerLogo.setSizeFull();
160        containerLogo.setStyleName("containerLogo");
161        winConnexion.addComponent(containerLogo);
162
163        logo = new VerticalLayout();
```



```
164        logo.setStyleName("v-logo");
165        logo.setWidth("201px");
166        logo.setHeight("184px");
167        containerLogo.addComponent(logo);
168        containerLogo.setComponentAlignment(logo, Alignment.TOP_CENTER);
169
170        VerticalLayout loginForm = getLoginForm();
171        winConnexion.addComponent(loginForm);
172
173        return winConnexion;
174    }
175
176    /**
177     * Construction du formulaire de connexion à l'interface. Ce layout est
178     * intégré à la fenêtre flottante
179     *
180     * @return Layout contenant le formulaire de connexion
181     */
182    private VerticalLayout getLoginForm() {
183        // VerticalLayout qui va contenir le TextField et le bouton
184        velConnexion = new VerticalLayout();
185        velConnexion.setSizeFull();
186        velConnexion.setMargin(true);
187        velConnexion.setSpacing(true);
188        velConnexion.setStyleName("velConnexion");
189
190        // TextField pour l'adresse mail
191        txtIdentifiant = new TextField();
192        txtIdentifiant.setCaption(Global.i("CAPTION_ADRESSE_EMAIL"));
193        txtIdentifiant.setImmediate(true);
194        txtIdentifiant.setWidth("230px");
195        txtIdentifiant.setRequired(true);
196        txtIdentifiant.setInputPrompt(Global.i("CAPTION_TYPE_YOUR_MAIL"));
197        txtIdentifiant.setStyleName("align-center");
198        txtIdentifiant.addValidator(new EmailValidator(Global
199                .i("CAPTION_EMAIL_VALIDATOR")));
200        txtIdentifiant.setValue("dominique.roduit@avs.vs.ch");
201        txtIdentifiant.addListener(new FieldEvents.BlurListener() {
202            @Override
203            public void blur(BlurEvent event) {
204                if (txtIdentifiant != null && txtIdentifiant.isValid()) {
205                    String email = txtIdentifiant.getValue().toString();
206
207                    // On vérifie si l'utilisateur existe déjà
208                    if (email != null) {
209                        ResultSet user = Sql.query(tbl_users
210                                .getUserByEmail(email));
211                        if (user != null) {
212                            existingUser = false;
213                            try {
214                                // L'utilisateur existe dans la base de données
215                                if (user.first()) {
216                                    existingUser = true;
217                                }
218                            } catch (SQLException e) {
219                                e.printStackTrace();
220                            }
221
222                            // Si l'utilisateur n'existe pas, on change le
223                            // bouton par "Inscription"
224                            if (!existingUser) {
225                                btnConn.setCaption(Global
```

```
                                .i("CAPTION_INSCRIPTION"));
                        } else {
                            btnConn.setCaption(Global
                                .i("CAPTION_CONNEXION"));
                        }
                        btnConn.setVisible(true);
                    }
                }
            } else {
                btnConn.setVisible(false);
            }
        }
    });

    velConnexion.addComponent(txtIdentifiant);
    velConnexion.setComponentAlignment(txtIdentifiant,
            Alignment.BOTTOM_CENTER);

    txtPass = new PasswordField();
    txtPass.setCaption(Global.i("CAPTION_PASSWORD"));
    txtPass.setWidth("230px");
    txtPass.setRequired(true);
    txtPass.setStyleName("align-center");
    txtPass.addValidator(new StringLengthValidator(Global
            .i("CAPTION_PASS_VALIDATOR"), Integer.parseInt(Global
            .getParm("PASS_MIN_LENGTH")), Integer.parseInt(Global
            .getParm("PASS_MAX_LENGTH")), false));
    txtPass.setRequiredError(Global.i("CAPTION_REQUIRE_ERROR_PASS"));
    velConnexion.addComponent(txtPass);
    velConnexion.setComponentAlignment(txtPass, Alignment.BOTTOM_CENTER);

    // Bouton de connexion pour la validation de l'adresse mail
    btnConn = new Button();
    btnConn.setCaption(Global.i("CAPTION_CONNEXION"));
    btnConn.setImmediate(true);
    btnConn.setVisible(false);
    velConnexion.addComponent(btnConn);
    velConnexion.setComponentAlignment(btnConn, Alignment.BOTTOM_CENTER);

    txtIdentifiant.focus();
    // Appel de l'évenement onclick sur le clique de la touche [ENTER]
    btnConn.setClickShortcut(KeyCode.ENTER);
    btnConn.addListener(connexionListener);

    return velConnexion;
}

/**
 * Retourne le layout racine du composant
 *
 * @return Layout racine (CompositionRoot)
 */
public HorizontalLayout getMainLayout() {
    return middleLayout;
}

/**
 * Retourne le bouton de connexion
 *
 * @return Bouton de connexion du formulaire
 */
public Button getButtonConnexion() {
```



```
    return btnConn;
}

/**
 * Retourne le textField d'identification
 *
 * @return TextField d'identification
 */
public TextField getTextFieldID() {
    return txtIdentifiant;
}

/**
 * Action exécuté sur le clique du bouton de connexion/inscription ou à la
 * validation du formulaire par le clavier
 */
private ClickListener connexionListener = new ClickListener() {
    @Override
    public void buttonClick(ClickEvent event) {
        System.out.println(Utilities.getCurrentDate() + " -- [Event] "
                + event.getButton().getCaption());

        String email = txtIdentifiant.getValue().toString();
        String pass = txtPass.getValue().toString();
        String passCrypted = SHA256.getHashValue(pass);
        boolean isInternal = Utilities.isInternal(email);
        String action = "";
        String message = email;
        String messDisplay = "";
        String notifText = "";
        Notification notif = null;

        message += (isInternal) ? " (interne) " : " (externe) ";

        // Si l'adresse et le mot de passe sont valides
        if (txtIdentifiant.isValid() && txtPass.isValid()) {
            Mailer.EmailValidator emailValidator = new Mailer.EmailValidator();

            // Si l'adresse e-mail entrée est une adresse avec un format
            // valide
            if (txtIdentifiant.isValid()) {
                // On vérifie si l'utilisateur existe déjà
                ResultSet user = Sql.query(tbl_users.getUserByEmail(email));
                existingUser = false;

                try {
                    if (user.first()) {
                        // L'utilisateur existe dans la base de données
                        existingUser = user.first();

                        // Si le mot de passe correspond à cet adresse
                        // e-mail
                        if (passCrypted.equals(user.getString("user_pass"))) {
                            if (user.getBoolean("user_validation")) { // Si
                                                                      //
  l'adresse
                                                                      // est
                                                                      // déjà
                                                                      //
  validée
                                action = "con_success";
                                message += " a accès, il a entré un bon mot de
```

```
   passe et son adresse est validée, on le connecte";
348
349                                 // Création de la session
350                                 UserSession.setID(user.getInt("PKNoUser"));
351                                 UserSession.setInternal(user
352                                         .getBoolean("user_internal"));
353                                 UserSession.setMail(user
354                                         .getString("user_mail"));
355                                 UserSession.setLangue(user
356                                         .getString("user_langue"));
357                                 UserSession.setAdmin(user
358                                         .getBoolean("user_admin"));
359
360                                 System.out.println("ID utilisateur = "
361                                         + UserSession.getID() + ", "
362                                         + UserSession.getMail());
363
364                                 setConnexionState(true);
365                             } else { // L'adresse n'est pas encore validée
366
367                                 action = "con_no_validate";
368                                 message += " a entré un bon mot de passe mais
   n'a pas encore validé son adresse, on lui renvoie un mail avec le lien sécurisé";
369                                 // Renvoi du mail contenant le lien sécurisé
370                                 Mailer.sendAccountValidationMail(email);
371
372                                 messDisplay += (!Utilities.getInternalName(
373                                         email).equals("")) ? "<b>"
374                                         + Global.i("CAPTION_HELLO") + ", "
375                                         + Utilities.getInternalName(email)
376                                         + "</b><br>" : "";
377                                 messDisplay += Global.i(
378                                         "CAPTION_EMAIL_REVALIDATION_SENT")
379                                         .replace("%email%", email);
380
381                                 // Affichage comme quoi le mail a été
382                                 // ré-envoyé
383                                 changeLogoArea(messDisplay,
384                                         Global.PATH_THEME_RESSOURCES
385                                                 + "signUpMailSend.png");
386
387                                 int nbEssai = tbl_journal_con
388                                         .getNumberOfConnexionsWhithoutValidatio
   n(email);
389                                 if (nbEssai > 5)
390                                     notifText = Global
391                                             .i("CAPTION_TRY_NUMBER")
392                                             + nbEssai
393                                             + ". "
394                                             +
   Global.i("CAPTION_IF_ERROR_OCCURED_MAIL_ME");
395
396                                 }
397                             } else {
398                                 action = "con_wrong_pass";
399                                 message += " existe mais a entré un mauvais mot de
   passe.";
400                                 notifText = Global.i("CAPTION_PASSWORD_ERROR");
401                             }
402
403                         } else {
404                             // L'utilisateur n'existe pas dans la base de
```



```
405                             // données, on lui envoye un mail avec un lien de
406                             // validation
407                             action = "con_new_user";
408                             message += " n'existe pas encore dans la base de
   données, on lui envoie un mail avec un lien sécurisé";
409
410                             messDisplay += (!Utilities.getInternalName(email)
411                                     .equals("")) ? "<b>"
412                                     + Global.i("CAPTION_WELCOME") + ", "
413                                     + Utilities.getInternalName(email)
414                                     + "</b><br>" : "";
415                             messDisplay += Global.i(
416                                     "CAPTION_EMAIL_VALIDATION_SENT").replace(
417                                     "%email%", email);
418
419                             // Inscription dans la base de données
420                             Sql.exec(tbl_users.getInsertQuery(email,
421                                     passCrypted, isInternal));
422
423                             // Envoi d'un mail contenant un lien de validation
424                             Mailer.sendAccountValidationMail(email);
425
426                             // Affichage comme quoi le mail a été envoyé
427                             changeLogoArea(messDisplay,
428                                     Global.PATH_THEME_RESSOURCES
429                                             + "signUpMailSend.png");
430                         }
431                     } catch (SQLException e) {
432                         action = "con_error_exception";
433                         message += "a déclenché une exception : erreur de connexion
   à la base de données\nPass : "
434                                 + Utilities.formatSQL(pass);
435                         notifText = Global.i("CAPTION_UNKNOW_ERROR");
436                         e.printStackTrace();
437                     }
438
439                 } else {
440                     action = "con_invalid_mail";
441                     message += " a tenté de se connecter en entrant une adresse
   e-mail invalide";
442                 }
443
444             } else {
445                 action = "con_invalid_infos";
446                 message += " a tenté de se connecter en entrant des informations
   invalides.";
447             }
448
449             if (email != null && action != null && message != null) {
450                 Sql.exec(tbl_journal_con.getInsertQuery(email, action, message));
451             }
452
453             // Affichage d'un notification s'il en existe une
454             if (!notifText.equals("")) {
455                 notif = new Notification(null, notifText,
456                         Notification.TYPE_ERROR_MESSAGE);
457                 notif.setPosition(Notification.POSITION_CENTERED_BOTTOM);
458                 notif.setDelayMsec(2000);
459                 getWindow().showNotification(notif);
460             }
461
462             // Déconnexion de la base de données
```

```
463             Sql.Disconnect();
464         }
465
466     };
467
468     /**
469      * Set l'état de la connexion
470      *
471      * @param connexionState
472      *            Etat de la connexion
473      */
474     public void setConnexionState(Boolean connexionState) {
475         this.connexionState = connexionState;
476     }
477
478     /**
479      * Retourne l'état de la connexion
480      *
481      * @return true si la connexion est ok
482      */
483     public Boolean getConnexionState() {
484         return connexionState;
485     }
486
487     /**
488      * Modifie la partie qui contient initialement le logo de la caisse en la
489      * chargeant par d'autres contenus
490      *
491      * @param content
492      *            Texte affiché
493      * @param icon
494      *            Icone d'illustration du texte
495      */
496     private void changeLogoArea(String content, String icon) {
497         // On redimensionne la fenêtre
498         winConnexion.setHeight("310px");
499         winConnexion.setWidth("440px");
500
501         // On cache le formulaire de connexion
502         velConnexion.setVisible(false);
503
504         // On supprime le logo
505         logo.removeStyleName("v-logo");
506         logo.setWidth("290px");
507         logo.setHeight("118px");
508
509         // On ajoute une image d'illustration
510         Embedded imgIllustration = new Embedded(null, new ThemeResource(icon));
511         logo.addComponent(imgIllustration);
512         logo.setComponentAlignment(imgIllustration, Alignment.TOP_CENTER);
513
514         // On ajoute le texte en dessous de l'illustration
515         Label label = new Label(content);
516         label.setContentMode(Label.CONTENT_XHTML);
517         label.setStyleName("align-center");
518         label.setWidth("400px");
519         label.setHeight("122px");
520         winConnexion.addComponent(label);
521     }
522
523 }
524
```



```
1 package main;
2
3 import sql.query.tbl_users;
21
22 /**
23  * <b>IMPORTANT !<b><br>
24  * <br>
25  * Construction du composant principal de l'application.<br>
26  * <br>
27  * Ce composant personnalisé est un layout qui contient l'interface utilisateur.
28  * Il est complexe et composé de plusieurs parties distinctes (Haut, Milieu,
29  * Bas). Chaque partie est dotée d'un menu. Cette class ne contient que les
30  * blocs et dispositions, le contenu est chargé depuis la class principale
31  * {@link CccvsTransfert}.
32  *
33  * @author Dominique Roduit
34  * @version 1.0.0
35  * @since 2013-03-01
36  *
37  */
38 public class MainLayout extends CustomComponent {
39
40     /** Hauteur du header **/
41     private final String HEADER_HEIGHT = Global
42             .getParm("LAYOUT_MAIN_HEADER_HEIGHT");
43     /** Hauteur du ruban **/
44     private final String RIBBON_HEIGHT = Global
45             .getParm("LAYOUT_MAIN_RIBBON_HEIGHT");
46     /** Largeur du menu principal **/
47     private final String MENU_WIDTH = Global.getParm("LAYOUT_MAIN_MENU_WIDTH");
48     /** Hauteur du footer **/
49     private final String FOOTER_HEIGHT = Global
50             .getParm("LAYOUT_MAIN_FOOTER_HEIGHT");
51     /**
52      * Layout principal pleine page contenant les 3 niveaux de bases (top,
53      * middle, bottom)
54      **/
55     private AbsoluteLayout mainLayout;
56
57     // Zone du haut
58     // ------------------------------------------------------------------
59     /** Layout général du haut **/
60     private HorizontalLayout topLayout;
61     /** Layout d'affichage du Logo de l'application **/
62     private HorizontalLayout headerLogo;
63     /** Layout du menu de droite dans le header de l'application **/
64     private HorizontalLayout headerMenu;
65     // ------------------------------------------------------------------------------
  -----
66
67     // Zone du milieu qui contient le menu + le corps de page
68     // -------------------------
69     /** Layout général du milieu **/
70     private HorizontalLayout middleLayout;
71     /** Contient les layout du menu **/
72     private VerticalLayout menu;
73     /** Ruban en haut du menu **/
74     private HorizontalLayout menuRibbon;
75     /** Zone dynamique du menu, placée en dessous du ruban **/
76     private VerticalLayout menuContent;
77
78     /**
```

```
     * Corps de page localisé dans la zone du milieu et qui contient les
     * éléments de la zone dynamique (rubban du body + zone de contenu)
     **/
    private VerticalLayout body;
    /** Ruban en haut du corps de page **/
    private HorizontalLayout bodyRibbon;
    /** Partie dynamique du body qui contiendra le contenu des pages **/
    private VerticalLayout bodyContent;
    /** Zone contenant le menu dynamique du ruban du corps de page **/
    private HorizontalLayout bodyRibbonMenu;
    // ------------------------------------------------------------------------
    // Zone du bas
    // ------------------------------------------------------------------------
    /** Layout général du bas **/
    private VerticalLayout bottomLayout;
    /**
     * Zone de téléchargement du footer<br>
     * Masquée par défaut
     **/
    private VerticalLayout bottomContentArea;
    /** Bouton de changement de langue **/
    private Button btChangeLanguage;

    // ------------------------------------------------------------------------

    /**
     * Assemblage des layouts par l'appel des méthodes de création des
     * composants
     */
    public MainLayout() {
        super.setSizeFull();

        // Absolute Layout qui va contenir tout les autres layout
        mainLayout = new AbsoluteLayout();
        mainLayout.setSizeFull();

        // Grid pleine page qui permet de placer les 3 premiers niveau de Layout
        // Verticalement
        GridLayout vMain = new GridLayout(1, 3);
        vMain.setSizeFull();
        vMain.setRowExpandRatio(1, 100);
        mainLayout.addComponent(vMain);

        // Header
        topLayout = new HorizontalLayout();
        topLayout.setHeight(HEADER_HEIGHT);
        topLayout.setWidth("100%");
        topLayout.setStyleName("v-header");
        vMain.addComponent(topLayout, 0, 0);
        buildTopLayout();

        // Body
        middleLayout = new HorizontalLayout();
        middleLayout.setSizeFull();
        middleLayout.setStyleName("v-body");
        vMain.addComponent(middleLayout, 0, 1);
        buildMiddleLayout();

        // Footer
```



```
        bottomLayout = new VerticalLayout();
        bottomLayout.setSizeFull();
        bottomLayout.setHeight(FOOTER_HEIGHT);
        bottomLayout.setStyleName("v-footer");
        vMain.addComponent(bottomLayout, 0, 2);
        buildBottomLayout();

        setCompositionRoot(mainLayout);
    }

    // ------------------------------------------------------------------------
    // ================ Construction du Layout du haut
    // ====================================
    // ------------------------------------------------------------------------
    /**
     * Construction du layout du haut de la page (header)
     */
    private void buildTopLayout() {
        // Grid qui va contenir le logo et le menu du header
        GridLayout grdTopLayout = new GridLayout(2, 1);
        grdTopLayout.setSizeFull();
        grdTopLayout.setColumnExpandRatio(1, 100);
        topLayout.addComponent(grdTopLayout);

        // Logo du header à la meme taille que le menu
        headerLogo = new HorizontalLayout();
        headerLogo.setSizeUndefined();
        headerLogo.setStyleName("v-header-logo");
        headerLogo.setMargin(false, false, false, true);
        grdTopLayout.addComponent(headerLogo, 0, 0);
        grdTopLayout.setComponentAlignment(headerLogo, Alignment.MIDDLE_LEFT);

        // Menu du header
        headerMenu = new HorizontalLayout();
        headerMenu.setSizeUndefined();
        headerMenu.setStyleName("v-header-menu");
        headerMenu.setSpacing(true);
        headerMenu.setMargin(false, true, false, false);
        grdTopLayout.addComponent(headerMenu, 1, 0);
        grdTopLayout.setComponentAlignment(headerMenu, Alignment.MIDDLE_RIGHT);
    }

    // ------------------------------------------------------------------------

    // ------------------------------------------------------------------------
    // ================ Construction du Layout du milieu
    // ====================================
    // ------------------------------------------------------------------------
    /**
     * Construction du Layout du millieu (menu et corps de page)
     */
    private void buildMiddleLayout() {

        GridLayout grdMiddleLayout = new GridLayout(2, 1);
        grdMiddleLayout.setSizeFull();
        grdMiddleLayout.setColumnExpandRatio(1, 100);
        middleLayout.addComponent(grdMiddleLayout);
```

```
196
197         // Menu qui contiendra l'arbre
198         menu = new VerticalLayout();
199         menu.setWidth(MENU_WIDTH);
200         menu.setHeight("100%");
201         menu.setStyleName("v-body-menu");
202         grdMiddleLayout.addComponent(menu, 0, 0);
203
204         // Fabrication du menu
205         loadMenuLayout();
206
207         // Corps de page qui contiendra le contenu principal
208         body = new VerticalLayout();
209         body.setSizeFull();
210         body.setStyleName("v-body");
211         grdMiddleLayout.addComponent(body, 1, 0);
212
213         // Fabrication du body
214         loadBodyLayout();
215     }
216
217     /**
218      * Construction des éléments du menu latéral
219      */
220     private void loadMenuLayout() {
221         // Grid pour utiliser le 100% de la hauteur du menu
222         GridLayout grdMenuLayout = new GridLayout(1, 2);
223         grdMenuLayout.setSizeFull();
224         grdMenuLayout.setRowExpandRatio(1, 100);
225         menu.addComponent(grdMenuLayout);
226
227         // Ruban en haut du menu
228         menuRibbon = new HorizontalLayout();
229         menuRibbon.setHeight(RIBBON_HEIGHT);
230         menuRibbon.setWidth("100%");
231         menuRibbon.setStyleName("v-menu-ribbon");
232         grdMenuLayout.addComponent(menuRibbon, 0, 0);
233
234         // Ajout du contenu du ruban
235         // -------------------------------------------
236
237         // Insertion du texte
238         Label lblFileManager = new Label(Global.getParm("APP_NAME")
239                 + " "
240                 + Global.getParm("APP_VERSION").substring(0,
241                         Global.getParm("APP_VERSION").lastIndexOf(".")));
242         lblFileManager.setStyleName("v-menu-ribbon-lbl");
243         menuRibbon.addComponent(lblFileManager);
244         // ------------------------------------------------------------------------
245
246         // Contenu du menu, zone dynamique qui contiendra l'arbre
247         menuContent = new VerticalLayout();
248         menuContent.setSizeFull();
249         menuContent.setSpacing(false);
250         menuContent.setStyleName("v-menu-content");
251         grdMenuLayout.addComponent(menuContent, 0, 1);
252     }
253
254     /**
255      * Construction des éléments du corps de page
256      */
257     private void loadBodyLayout() {
```



```
258         // Grid pour utiliser le 100% de la largeur du body
259         GridLayout grdBodyLayout = new GridLayout(1, 2);
260         grdBodyLayout.setSizeFull();
261         grdBodyLayout.setRowExpandRatio(1, 100);
262         body.addComponent(grdBodyLayout);
263         body.setExpandRatio(grdBodyLayout, 100);
264
265         // Ruban en haut du corps de page qui contiendra les boutons
266         bodyRibbon = new HorizontalLayout();
267         bodyRibbon.setHeight(RIBBON_HEIGHT);
268         bodyRibbon.setWidth("100%");
269         bodyRibbon.setStyleName("v-body-ribbon");
270         grdBodyLayout.addComponent(bodyRibbon, 0, 0);
271
272         // Ajout d'un layout horizontal aligné a gauche pour le boutons
273         // d'actions
274         bodyRibbonMenu = new HorizontalLayout();
275         bodyRibbonMenu.setStyleName("v-body-ribbon-menu");
276         bodyRibbonMenu.setSpacing(true);
277         bodyRibbon.addComponent(bodyRibbonMenu);
278         bodyRibbon.setComponentAlignment(bodyRibbonMenu, Alignment.MIDDLE_LEFT);
279
280         // Partie dynamique du body qui contiendra vraiment le contenu des
281         // différentes pages
282         bodyContent = new VerticalLayout();
283         bodyContent.setSizeFull();
284         bodyContent.setStyleName("v-body");
285         grdBodyLayout.addComponent(bodyContent);
286     }
287
288     // ------------------------------------------------------------------------
    ---------
289
290     // ------------------------------------------------------------------------
    ---------
291     // ================== Construction du Layout du bas
292     // ====================================
293     // ------------------------------------------------------------------------
    ---------
294     /**
295      * Construction du layout du bas (footer)
296      */
297     public void buildBottomLayout() {
298         GridLayout grdBottomLayout = new GridLayout(1, 2);
299         grdBottomLayout.setSizeFull();
300         grdBottomLayout.setRowExpandRatio(1, 100);
301         bottomLayout.addComponent(grdBottomLayout);
302
303         // Zone en haut du layout pour rétrécir le layout
304         HorizontalLayout footerToogle = new HorizontalLayout();
305         footerToogle.setWidth("100%");
306         footerToogle.setHeight("25px");
307         footerToogle.setStyleName("v-footer-toogle");
308         grdBottomLayout.addComponent(footerToogle, 0, 0);
309
310         // Contenu de la zone toogle ---------------------------------------
311         GridLayout grdToogle = new GridLayout(2, 1);
312         grdToogle.setSizeFull();
313         footerToogle.addComponent(grdToogle);
314
315         // Insertion du Label a coté de l'image de transfert
316         Label lblUserMail = new Label();
```

```
317         lblUserMail.setIcon(new ThemeResource(Global.IMG_APPROVED));
318         lblUserMail.setCaption(UserSession.getMail());
319         lblUserMail.setStyleName("v-footer-toogle-text");
320         grdToogle.addComponent(lblUserMail, 0, 0);
321         grdToogle.setComponentAlignment(lblUserMail, Alignment.MIDDLE_LEFT);
322 
323         bottomLayout.setHeight("25px");
324 
325         // Insertion de l'image pour rétrécir le footer
326         btChangeLanguage = new Button("", new Button.ClickListener() {
327             @Override
328             public void buttonClick(ClickEvent event) {
329                 String newLanguage = (Global.IMG_FLAG_COUNTRY.indexOf("de") > -1) ? "fr"
330                         : "de";
331                 Global.IMG_FLAG_COUNTRY = Global.PATH_THEME_RESSOURCES
332                         + newLanguage + ".png";
333                 event.getComponent().setIcon(
334                         new ThemeResource(Global.IMG_FLAG_COUNTRY));
335 
336                 UserSession.setLangue(newLanguage);
337                 // Mise à jour de la langue de la base de données
338                 if (UserSession.getID() > 0) {
339                     Sql.exec(tbl_users.getUpdLanguage(newLanguage));
340                 }
341                 // Rechargement du tableau des traductions
342                 Global.reloadTranslations();
343                 // Rechargement du layout complet depuis la class principal
344                 // CccvsTransfert
345             }
346         });
347         btChangeLanguage.setDescription(Global.i("CAPTION_CHANGE_LANGUAGE"));
348         btChangeLanguage.setIcon(new ThemeResource(Global.IMG_FLAG_COUNTRY));
349         btChangeLanguage.setStyleName("v-footer-toogle-button");
350         grdToogle.addComponent(btChangeLanguage, 1, 0);
351         grdToogle.setComponentAlignment(btChangeLanguage,
352                 Alignment.MIDDLE_RIGHT);
353 
354         // -----------------------------------------------------------------
355 
356         // Zone qui contiendra les téléchargements en cours
357         bottomContentArea = new VerticalLayout();
358         bottomContentArea.setSizeFull();
359         grdBottomLayout.addComponent(bottomContentArea, 0, 1);
360 
361     }
362 
363     // ------------------------------------------------------------------------
    ---------
364     /**
365      * Affiche/Masque le footer
366      *
367      * @param enabled
368      *            true : affiché, false : désactivé
369      */
370     public void setFooterEnabled(boolean enabled) {
371         bottomContentArea.setVisible(enabled);
372         int size = (enabled) ? 190 : 25;
373         bottomLayout.setHeight(size + "px");
374     }
375 
376     /**
```



```
377      * Restaure la taille initiale du ruban
378      */
379     public void restoreRibbonHeight() {
380         bodyRibbon.setHeight(RIBBON_HEIGHT);
381     }
382 
383     // ------------------------------------------------------------------------
    ---------
384     // ==================== getters des composants à contenu dynamique
385     // ====================
386     // ------------------------------------------------------------------------
    ---------
387     /**
388      * Retourne la zone du header contenant le logo
389      *
390      * @return Zone du header contenant le logo
391      */
392     public HorizontalLayout getHeaderLogo() {
393         return headerLogo;
394     }
395 
396     /**
397      * Retourne le menu du header
398      *
399      * @return Menu du header
400      */
401     public HorizontalLayout getHeaderMenu() {
402         return headerMenu;
403     }
404 
405     /**
406      * Retourne le corps de page dans lequel on va insérer notre contenu
407      *
408      * @return Corps de page
409      */
410     public VerticalLayout getBody() {
411         return bodyContent;
412     }
413 
414     /**
415      * @return Zone racine du ruban du corps de page
416      */
417     public HorizontalLayout getBodyRibbonWrapper() {
418         return bodyRibbon;
419     }
420 
421     /**
422      * Retourne le ruban du corps de page
423      *
424      * @return Ruban du corps de page
425      */
426     public HorizontalLayout getBodyRibbon() {
427         return bodyRibbonMenu;
428     }
429 
430     /**
431      * Retourne la zone d'affichage dynamique du menu
432      *
433      * @return Menu latéral
434      */
435     public VerticalLayout getMenu() {
436         return menuContent;
```

```
437     }
438
439     /**
440      * Retourne le ruban du menu
441      *
442      * @return Ruban du menu
443      */
444     public HorizontalLayout getMenuRibbon() {
445         return menuRibbon;
446     }
447
448     /**
449      * Retourne la zone de téléchargement dans le footer
450      *
451      * @return Zone de téléchargement du footer
452      */
453     public VerticalLayout getFooter() {
454         return bottomContentArea;
455     }
456
457     /**
458      * Retourne le bouton de changement de langue
459      *
460      * @return Bouton de changement de langue
461      */
462     public Button getBtChangeLanguage() {
463         return btChangeLanguage;
464     }
465
466     // ------------------------------------------------------------------------
    ---------
467
468     // ------------------------------------------------------------------------
    ---------
469     // ==================== setters des composants à contenu dynamique
470     // ====================
471     // ------------------------------------------------------------------------
    ---------
472     /**
473      * Fixe la taille du menu latéral
474      *
475      * @param width
476      *            Taille du menu en px
477      */
478     public void setMenuWidth(String width) {
479         menu.setWidth(width);
480     }
481     // ------------------------------------------------------------------------
    ---------
482 }
483
```

PACKAGE

MODEL

- ActionJournalFold
- Contact
- Files
- Folder
- JournalCon
- User

Model des données de chaque table de la base de données. Stockage sous forme d'objets dans le but de pouvoir récupérer tous les attributs simplement.

```java
package model;

/**
 * Modèle pour le stockage d'actions journalisées sur les dossiers<br>
 * <b>Table :</b> trans_journal_folders
 *
 * @author Dominique Roduit
 *
 */
public class ActionJournalFold {
    /** Clé primaire de l'action **/
    private int PKNoJourFolder;
    /**
     * Type de l'action journalisée<br>
     * <br>
     * <b>Actions</b><br>
     * <li>download</li><li>consult</li>
     **/
    private String action_type = "";
    /** Description de l'action exécutée **/
    private String action_description = "";
    /** Fichier concerné par l'action **/
    private Files action_file;
    /** Dossier concerné par l'action **/
    private Folder action_folder;
    /** Date de l'action **/
    private String action_date;
    /** Contact qui déclenche l'action **/
    private Contact action_contact;

    /**
     * Création d'un objet pour une action journalisée exécutée sur un dossier
     *
     * @param Type
     *            Type d'action
     * @param Description
     *            Description de l'action
     */
    public ActionJournalFold(String Type, String Description) {
        setAction_type(Type);
        setAction_description(Description);
    }

    /**
     * Création d'un objet pour une action journalisée exécutée sur un dossier
     *
     * @param action_type
     *            Type d'action
     * @param action_description
     *            Description de l'action
     * @param action_file
     *            Fichier concerné par l'action
     * @param action_folder
     *            Dossier concerné par l'action
     * @param action_date
     *            Date de l'action
     * @param action_contact
     *            Contact qui déclenche l'action
     */
    public ActionJournalFold(int PKNoJourFolder, String action_type,
            Files action_file, Folder action_folder, String action_date,
            Contact action_contact) {
```



```java
        this.PKNoJourFolder = PKNoJourFolder;
        this.action_type = action_type;
        this.action_file = action_file;
        this.action_folder = action_folder;
        this.action_date = action_date;
        this.action_contact = action_contact;
    }

    /**
     * Obtention du type d'action
     *
     * @return Type d'action
     */
    public String getAction_type() {
        return action_type;
    }

    /**
     * Enregistrement du type d'action
     *
     * @param action_type
     *            Type d'action
     */
    public void setAction_type(String action_type) {
        this.action_type = action_type;
    }

    /**
     * Obtention de la description de l'action effectuée
     *
     * @return Description de l'action effectuée
     */
    public String getAction_description() {
        return action_description;
    }

    /**
     * Enregistrement de la description de l'action effectuée
     *
     * @param action_description
     *            Description de l'action effectuée
     */
    public void setAction_description(String action_description) {
        this.action_description = action_description;
    }

    /**
     * @return Fichier concerné par l'action
     */
    public Files getAction_file() {
        return action_file;
    }

    /**
     * @param action_file
     *            Fichier concerné par l'action
     */
    public void setAction_file(Files action_file) {
        this.action_file = action_file;
    }

    /**
```

```
     * @return Dossier concerné par l'action
     */
    public Folder getAction_folder() {
        return action_folder;
    }

    /**
     * @param action_folder
     *            Dossier concerné par l'action
     */
    public void setAction_folder(Folder action_folder) {
        this.action_folder = action_folder;
    }

    /**
     * @return Date de l'action
     */
    public String getAction_date() {
        return action_date;
    }

    /**
     * @param action_date
     *            Date de l'action
     */
    public void setAction_date(String action_date) {
        this.action_date = action_date;
    }

    /**
     * @return Contact qui déclenche l'action
     */
    public Contact getAction_contact() {
        return action_contact;
    }

    /**
     * @param action_contact
     *            Contact qui déclenche l'action
     */
    public void setAction_contact(Contact action_contact) {
        this.action_contact = action_contact;
    }

    /**
     * @return Clé primaire de l'action
     */
    public int getPKNoJourFolder() {
        return PKNoJourFolder;
    }

    /**
     * @param pKNoJourFolder
     *            Clé primaire de l'action
     */
    public void setPKNoJourFolder(int pKNoJourFolder) {
        PKNoJourFolder = pKNoJourFolder;
    }

}
```



```
package model;

import java.io.Serializable;

/**
 * Modèle de contact.<br>
 * Cette class correspond aux champs de la base de données pour la table
 * tbl_contacts.
 *
 * @author Dominique Roduit
 *
 */
public class Contact implements Serializable {
    /** Clé primaire du contact **/
    private int PK = 0;
    /** Clé étrangère vers l'utilisateur qui a créé le contact **/
    private int FKNoUser = 0;
    /** Nom du contact **/
    private String name = "";
    /** Adresse e-mail du contact **/
    private String mail = "";
    /** Indique si le contact est un interne ou non **/
    private boolean internal = false;
    /** Date de création du contact **/
    private String creation_date = "";

    public Contact() {

    }

    /**
     * Constructeur du model pour le stockage d'informations sur un contact
     *
     * @param PK
     *            Clé primaire du contact
     * @param name
     *            Nom du contact
     * @param mail
     *            Adresse e-mail du contact
     * @param internal
     *            true si le contact est un interne
     */
    public Contact(int PK, String name, String mail, boolean internal) {
        setPK(PK);
        setName(name);
        setMail(mail);
        setInternal(internal);
    }

    /**
     * Contructeur du model pour le stockage d'informations sur un contact
     *
     * @param PK
     *            Clé primaire du contact
     * @param FKNoUser
     *            Clé étrangère vers l'utilisateur qui détient ce contact dans
     *            sa liste
     * @param name
     *            Nom du contact
     * @param mail
     *            Adresse e-mail du contact
     * @param creation_date
```

```
63      *            Date de créatin du contact
64      * @param internal
65      *            true si le contact est un interne
66      */
67     public Contact(int PKNoContact, int FKNoUser, String name, String mail,
68             String creation_date, boolean internal) {
69         this(PKNoContact, name, mail, internal);
70         setFKNoUser(FKNoUser);
71         setCreation_date(creation_date);
72     }
73
74     /**
75      * Obtention de la clé primaire du contact
76      *
77      * @return Cél primaire du contact
78      */
79     public int getPK() {
80         return PK;
81     }
82
83     /**
84      * Définition de la clé primaire du contact
85      *
86      * @param pK
87      *            Clé primaire du contact
88      */
89     public void setPK(int pK) {
90         PK = pK;
91     }
92
93     /**
94      * Obtention du nom du contact
95      *
96      * @return Nom du contact
97      */
98     public String getName() {
99         return name;
100    }
101
102    /**
103     * Définition du nom du contact
104     *
105     * @param name
106     *            Nom du contact
107     */
108    public void setName(String name) {
109        this.name = name;
110    }
111
112    /**
113     * Obtention de l'adresse e-mail du contact
114     *
115     * @return Adresse e-mail du contact
116     */
117    public String getMail() {
118        return mail;
119    }
120
121    /**
122     * Définition de l'adresse e-mail du contact
123     *
124     * @param email
```



```
125     *            Adresse e-mail du contact
126     */
127    public void setMail(String email) {
128        this.mail = email;
129    }
130
131    /**
132     * Vidage des champs pour l'annulation des valeurs entrées dans le
133     * formulaire d'ajout/édition
134     */
135    public void reset() {
136        setName("");
137        setMail("");
138    }
139
140    /**
141     * Obtention de la clé étrangère de l'utilisateur détenant le contact
142     *
143     * @return Clé étrangère de l'utilisateur
144     */
145    public int getFKNoUser() {
146        return FKNoUser;
147    }
148
149    /**
150     * Définition de la clé étrangère de l'utilisateur détenant le contact
151     *
152     * @param fKNoUser
153     *            Clé étrangère de l'utilisateur
154     */
155    public void setFKNoUser(int fKNoUser) {
156        FKNoUser = fKNoUser;
157    }
158
159    /**
160     * Indique si le contact est un interne ou non
161     *
162     * @return true si le contact est un interne
163     */
164    public boolean isInternal() {
165        return internal;
166    }
167
168    /**
169     * Définit si le contact est un interne ou non
170     *
171     * @param internal
172     *            true si le contact est un interne
173     */
174    public void setInternal(boolean internal) {
175        this.internal = internal;
176    }
177
178    /**
179     * @return Date de création du contact
180     */
181    public String getCreation_date() {
182        return creation_date;
183    }
184
185    /**
186     * @param creation_date
```

```
187      *            Date de création du contact
188      */
189     public void setCreation_date(String creation_date) {
190         this.creation_date = creation_date;
191     }
192 }
```



```
1 package model;
2
3 import global.Global;
7
8 /**
9  * Modèle d'un fichier (Table trans_files).<br>
10 * Permet le stockage des informations sur un fichier dans un objet.
11 *
12 * @author Dominique Roduit
13 *
14 */
15 public class Files {
16     /** Clé primaire du fichier **/
17     private int PKNoFile = 0;
18     /** Clé étrangère du dossier qui contient le fichier **/
19     private int FKNoFolder = 0;
20     /** Nom du fichier (tel qu'enregistré sur le disque) **/
21     private String file_name = "";
22     /** Nom que l'utilisateur a donné au fichier **/
23     private String file_rename = "";
24     /** Taille du fichier (en octet) **/
25     private long file_size = 0;
26     /** Extension du fichier **/
27     private String file_extension = "";
28     /** Description du fichier **/
29     private String file_description = "";
30     /** Taille du fichier formattée par la class Utilities. **/
31     private String file_size_formatted = "";
32
33     /**
34      * Constructeur du model pour le stockage d'un model de fichier
35      *
36      * @param PKNoFile
37      *            Clé primaire du fichier
38      * @param FKNoFolder
39      *            Clé étrangère du dossier
40      * @param file_name
41      *            Nom du fichier
42      * @param file_rename
43      *            Nom du fichier spécifié par l'utilisateur
44      * @param file_size
45      *            Taille du fichier
46      * @param file_extension
47      *            Extension du fichier
48      * @param file_description
49      *            Description du fichier
50      */
51     public Files(int PKNoFile, int FKNoFolder, String file_name,
52             String file_rename, long file_size, String file_extension,
53             String file_description) {
54         setPKNoFile(PKNoFile);
55         setFKNoFolder(FKNoFolder);
56         setFile_name(file_name);
57         setFile_rename(file_rename);
58         setFile_size(file_size);
59         setFile_extension(file_extension);
60         setFile_description(file_description);
61     }
62
63     /**
64      * Constructeur du model pour le stockage d'un model de fichier
65      *
```

```
     * @param PKNoFile
     *            Clé primaire du fichier
     * @param FKNoFolder
     *            Clé étrangère du dossier
     * @param file_name
     *            Nom du fichier
     * @param file_rename
     *            Nom du fichier spécifié par l'utilisateur
     * @param file_size
     *            Taille du fichier
     * @param file_extension
     *            Extension du fichier
     * @param file_description
     *            Description du fichier
     * @param file_size_formatted
     *            Taille du fichier formattée
     */
    public Files(int PKNoFile, int FKNoFolder, String file_name,
            String file_rename, long file_size, String file_extension,
            String file_description, String file_size_formatted) {

        this(PKNoFile, FKNoFolder, file_name, file_rename, file_size,
                file_extension, file_description);
        setFile_size_formatted(file_size_formatted);
    }

    /**
     * Obtention de la clé primaire du fichier
     *
     * @return Clé primaire du fichier
     */
    public int getPKNoFile() {
        return PKNoFile;
    }

    /**
     * Fixation de la clé primaire du fichier
     *
     * @param pKNoFile
     *            Clé primaire du fichier
     */
    public void setPKNoFile(int pKNoFile) {
        PKNoFile = pKNoFile;
    }

    /**
     * Obtention de la clé étrangère vers le dossier contenant le fichier
     *
     * @return Clé étrangère vers le dossier
     */
    public int getFKNoFolder() {
        return FKNoFolder;
    }

    /**
     * Paramétrage de la clé étrangère
     *
     * @param fKNoFolder
     *            Clé étrangère vers le dossier
     */
    public void setFKNoFolder(int fKNoFolder) {
        FKNoFolder = fKNoFolder;
```



```
    }

    /**
     * Obtention du nom du fichier tel qu'il est enregistré sur le disque
     *
     * @return Nom du fichier sur le disque
     */
    public String getFile_name() {
        return file_name;
    }

    /**
     * Réglage du nom de fichier
     *
     * @param file_name
     *            Nom du fichier
     */
    public void setFile_name(String file_name) {
        this.file_name = file_name;
    }

    /**
     * Obtention du nom de fichier spécifié par l'utilisateur
     *
     * @return Nom du fichier spécifié par l'utilisateur
     */
    public String getFile_rename() {
        return file_rename;
    }

    /**
     * Réglage du nom du fichier spécifié par l'utilisateur
     *
     * @param file_rename
     *            Nom du fichier spécifié par l'utilisateur
     */
    public void setFile_rename(String file_rename) {
        this.file_rename = file_rename;
    }

    /**
     * Obtention de la taille du fichier
     *
     * @return Taille du fichier (Octets)
     */
    public long getFile_size() {
        return file_size;
    }

    /**
     * Réglage de la taille du fichier
     *
     * @param file_size
     *            Taille du fichier en octets
     */
    public void setFile_size(long file_size) {
        this.file_size = file_size;
    }

    /**
     * Obtention de l'extension du fichier
     *
```

```
     * @return Extension du fichier
     */
    public String getFile_extension() {
        return file_extension;
    }

    /**
     * Réglage de l'extension du fichier
     *
     * @param file_extension
     *            Extension du fichier
     */
    public void setFile_extension(String file_extension) {
        this.file_extension = file_extension;
    }

    /**
     * Obtention de la description du fichier
     *
     * @return Description du fichier
     */
    public String getFile_description() {
        return file_description;
    }

    /**
     * Réglage de la description du fichier
     *
     * @param file_description
     *            Description du fichier
     */
    public void setFile_description(String file_description) {
        this.file_description = file_description;
    }

    /**
     * Obtention de la taille du fichier formattée
     *
     * @return Taille formattée
     */
    public String getFile_size_formatted() {
        return file_size_formatted;
    }

    /**
     * Réglage de la taille formattée du fichier
     *
     * @param file_size_formatted
     *            Taille formattée du fichier
     */
    public void setFile_size_formatted(String file_size_formatted) {
        this.file_size_formatted = file_size_formatted;
    }
}
```



```
package model;

import com.vaadin.ui.Label;

/**
 * Modèle d'un dossier de transfert (Table trans_folders).<br>
 * Permet le stockage des informations sur un dossier dans un objet.
 *
 * @author Dominique Roduit
 *
 */
public class Folder {
    /** Clé primaire du dossier **/
    private int PKNoFolder;
    /** Clé étrangère vers l'utilisateur, auteur du dossier */
    private int FKNoUser;
    /** Nom du dossier **/
    private String name = "";
    /** Description du dossier **/
    private String description = "";
    /** Date de création du dossier **/
    private String creation_date;
    /** Durée de vie du dossier (en jours) **/
    private double expiration = 1;
    /** Date d'expiration du dossier **/
    private String expiration_date = "";
    /** Statut du dossier (archive ou disponible) **/
    private boolean archive = false;

    public Folder() {

    }

    /**
     * Constructeur du model pour le stockage d'un model de dossier
     *
     * @param PKNoFolder
     *            Clé primaire du dossier
     * @param FKNoUser
     *            Clé étrangère vers l'utilisateur
     * @param folder_name
     *            Nom du dossier
     * @param folder_creation_date
     *            Date de création du dossier
     * @param folder_expiration
     *            Date d'expiration du dossier
     * @param folder_archive
     *            true si le dossier est une archive
     */
    public Folder(int PKNoFolder, int FKNoUser, String folder_name,
            String description, String folder_creation_date,
            String folder_expiration, boolean folder_archive) {
        setPKNoFolder(PKNoFolder);
        setFKNoUser(FKNoUser);
        setName(folder_name);
        setCreation_date(folder_creation_date);
        setExpiration_date(folder_expiration);
        setArchive(folder_archive);
        setDescription(description);
    }

    /**
```

```java
     * Obtention de la clé primaire du dossier
     *
     * @return Clé primaire du dossier
     */
    public int getPKNoFolder() {
        return PKNoFolder;
    }

    /**
     * Réglage de la clé primaire du dossier
     *
     * @param pKNoFolder
     *            Clé primaire du dossier
     */
    public void setPKNoFolder(int pKNoFolder) {
        PKNoFolder = pKNoFolder;
    }

    /**
     * Obtention de la clé étrangère de l'utilisateur
     *
     * @return Clé étrangère de l'utilisateur
     */
    public int getFKNoUser() {
        return FKNoUser;
    }

    /**
     * Réglage de la clé étrangère de l'utilisateur
     *
     * @param fKNoUser
     *            Clé étrangère de l'utilisateur
     */
    public void setFKNoUser(int fKNoUser) {
        FKNoUser = fKNoUser;
    }

    /**
     * Obtention du nom du dossier
     *
     * @return Nom du dossier
     */
    public String getName() {
        return name;
    }

    /**
     * Réglage du nom du dossier
     *
     * @param name
     *            Nom du dossier
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * @return La Description du dossier
     */
    public String getDescription() {
        return description;
    }
```



```java

    /**
     * @param description
     *            Description du dossier
     */
    public void setDescription(String description) {
        this.description = description;
    }

    /**
     * Obtention de la date de création du dossier
     *
     * @return Date de création du dossier
     */
    public String getCreation_date() {
        return creation_date;
    }

    /**
     * Définition de la date de création du dossier
     *
     * @param creation_date
     *            Date de création du dossier
     */
    public void setCreation_date(String creation_date) {
        this.creation_date = creation_date;
    }

    /**
     * Obtention de la durée de vie du dossier
     *
     * @return Durée de vie du dossier
     */
    public double getExpiration() {
        return expiration;
    }

    /**
     * Définition de la durée de vie du dossier
     *
     * @param expiration
     *            Durée de vie du dossier en jours
     */
    public void setExpiration(double expiration) {
        this.expiration = expiration;
    }

    /**
     * Indique si le dossier est une archive ou non
     *
     * @return true si le dossier est une archive (expiré)
     */
    public boolean getArchive() {
        return archive;
    }

    /**
     * Définit si le dossier est une archive ou non
     *
     * @param archive
     *            true si le dossier est une archive
     */
```

```
187     public void setArchive(boolean archive) {
188         this.archive = archive;
189     }
190 
191     /**
192      * Obtention de la date d'expiration du dossier
193      *
194      * @return Date d'expiration du dossier au format MySQL
195      */
196     public String getExpiration_date() {
197         return expiration_date;
198     }
199 
200     /**
201      * Définition de la date d'expiration du dossier
202      *
203      * @param expiration_date
204      *            Date d'expiration du dossier au format MySQL
205      */
206     public void setExpiration_date(String expiration_date) {
207         this.expiration_date = expiration_date;
208     }
209 }
210 
```



```
1 package model;
2 
3 /**
4  * Modèle d'une action de connexion journalisée (Table
5  * trans_journal_connections).<br>
6  * Permet le stockage des actions de connexion dans un objet Java.
7  *
8  * @author Dominique Roduit
9  *
10 */
11 public class JournalCon {
12     /** Clé primaire de l'action **/
13     private int PKNoJourConnection;
14     /** L'adresse e-mail de la personne qui déclenche l'action **/
15     private String joco_mail;
16     /** La date à laquelle l'action est survenue **/
17     private String joco_date;
18     /** L'adresse IP publique de la personne qui a déclenché l'action **/
19     private String joco_ip;
20     /** Système d'exploitation de la personne qui déclenche l'action **/
21     private String joco_os;
22     /** Nom de navigateur et version de la personne qui déclenche l'action **/
23     private String joco_browser;
24     /** Action déclenchée **/
25     private String joco_action;
26     /** Commentaire sur l'action **/
27     private String joco_comment;
28 
29     public JournalCon(int pKNoJourConnection, String joco_mail,
30             String joco_date, String joco_ip, String joco_os,
31             String joco_browser, String joco_action, String joco_comment) {
32         PKNoJourConnection = pKNoJourConnection;
33         this.joco_mail = joco_mail;
34         this.joco_date = joco_date;
35         this.joco_ip = joco_ip;
36         this.joco_os = joco_os;
37         this.joco_browser = joco_browser;
38         this.joco_action = joco_action;
39         this.joco_comment = joco_comment;
40     }
41 
42     /**
43      * @return Clé primaire de l'action
44      */
45     public int getPKNoJourConnection() {
46         return PKNoJourConnection;
47     }
48 
49     /**
50      * @param pKNoJourConnection
51      *            Clé primaire de l'action
52      */
53     public void setPKNoJourConnection(int pKNoJourConnection) {
54         PKNoJourConnection = pKNoJourConnection;
55     }
56 
57     /**
58      * @return L'adresse e-mail de la personne qui déclenche l'action
59      */
60     public String getJoco_mail() {
61         return joco_mail;
62     }
```

```java

    /**
     * @param joco_mail
     *            L'adresse e-mail de la personne qui déclenche l'action
     */
    public void setJoco_mail(String joco_mail) {
        this.joco_mail = joco_mail;
    }

    /**
     * @return La date à laquelle l'action est survenue
     */
    public String getJoco_date() {
        return joco_date;
    }

    /**
     * @param joco_date
     *            La date à laquelle l'action est survenue
     */
    public void setJoco_date(String joco_date) {
        this.joco_date = joco_date;
    }

    /**
     * @return L'adresse IP publique de la personne qui a déclenché l'action
     */
    public String getJoco_ip() {
        return joco_ip;
    }

    /**
     * @param joco_ip
     *            L'adresse IP publique de la personne qui a déclenché l'action
     */
    public void setJoco_ip(String joco_ip) {
        this.joco_ip = joco_ip;
    }

    /**
     * @return Système d'exploitation de la personne qui déclenche l'action
     */
    public String getJoco_os() {
        return joco_os;
    }

    /**
     * @param joco_os
     *            Système d'exploitation de la personne qui déclenche l'action
     */
    public void setJoco_os(String joco_os) {
        this.joco_os = joco_os;
    }

    /**
     * @return Nom de navigateur et version de la personne qui déclenche
     *         l'action
     */
    public String getJoco_browser() {
        return joco_browser;
    }

```



```java
    /**
     * @param joco_browser
     *            Nom de navigateur et version de la personne qui déclenche
     *            l'action
     */
    public void setJoco_browser(String joco_browser) {
        this.joco_browser = joco_browser;
    }

    /**
     * @return Action déclenchée
     */
    public String getJoco_action() {
        return joco_action;
    }

    /**
     * @param joco_action
     *            Action déclenchée
     */
    public void setJoco_action(String joco_action) {
        this.joco_action = joco_action;
    }

    /**
     * @return Commentaire sur l'action
     */
    public String getJoco_comment() {
        return joco_comment;
    }

    /**
     * @param joco_comment
     *            Commentaire sur l'action
     */
    public void setJoco_comment(String joco_comment) {
        this.joco_comment = joco_comment;
    }

}

```

```java
package model;

/**
 * Modèle d'un utilisateur (Table trans_users).<br>
 * Permet le stockage des informations sur un utilisateur dans un objet.
 *
 * @author Dominique Roduit
 *
 */
public class User {
    /** Clé primaire de l'utilisateur **/
    private int PKNoUser = 0;
    /** Adresse e-mail de l'utilisateur **/
    private String user_mail = "";
    /** Mot de passe de l'utilisateur (crypté) **/
    private String user_pass = "";
    /** Indique si l'utilisateur est interne ou non **/
    private boolean user_internal = false;
    /** Indique si l'utilisateur à validé son compte ou non **/
    private boolean user_validation = false;
    /** Contient la date de validation du compte **/
    private String user_validation_date = "";
    /** Contient la langue préférée de l'utilisateur **/
    private String user_langue = "";
    /** Indique si l'utilisateur est administrateur ou non **/
    private boolean user_admin = false;

    public User() {
    }

    /**
     * Création d'un objet, model d'utilisateur
     *
     * @param PKNoUser
     *            Clé primaire de l'utilisateur
     * @param user_mail
     *            Adresse e-mail de l'utilisateur
     * @param user_pass
     *            Mot de passe de l'utilisateur
     * @param user_internal
     *            Indique si l'utilisateur est interne ou pas
     * @param user_validation
     *            Indique si l'utilisateur a validé son compte
     * @param user_validation_date
     *            Date de validation du compte utilisateur
     * @param user_langue
     *            Langue de l'utilisateur
     * @param user_admin
     *            Indique si l'utilisateur est un administrateur
     */
    public User(int PKNoUser, String user_mail, String user_pass,
            boolean user_internal, boolean user_validation,
            String user_validation_date, String user_langue, boolean user_admin) {
        setPKNoUser(PKNoUser);
        setUser_mail(user_mail);
        setUser_pass(user_pass);
        setInternal(user_internal);
        setUser_validation(user_validation);
        setUser_validation_date(user_validation_date);
        setUser_langue(user_langue);
        setUser_admin(user_admin);
    }
```



```java

    /**
     * Obtention de la clé primaire
     *
     * @return Clé primaire
     */
    public int getPKNoUser() {
        return PKNoUser;
    }

    /**
     * Enregistrement de la clé primaire de l'utilisateur
     *
     * @param PKNoUser
     *            Clé primaire de l'utilisateur
     */
    public void setPKNoUser(int PKNoUser) {
        this.PKNoUser = PKNoUser;
    }

    /**
     * Obtention de l'adresse e-mail de l'utilisateur
     *
     * @return Adresse e-mail de l'utilisateur
     */
    public String getUser_mail() {
        return user_mail;
    }

    /**
     * Réglage de l'adresse e-mail de l'utilisateur
     *
     * @param user_mail
     */
    public void setUser_mail(String user_mail) {
        this.user_mail = user_mail;
    }

    /**
     * Obtention du mot de passe crypté de l'utilisateur
     *
     * @return Mot de passe crypté
     */
    public String getUser_pass() {
        return user_pass;
    }

    /**
     * Définition du mot de passe de l'utilisateur
     *
     * @param pass
     *            Mot de passe de l'utilisateur (crypté)
     */
    public void setUser_pass(String pass) {
        this.user_pass = pass;
    }

    /**
     * Obtention de l'état interne ou non de l'utilisateur
     *
     * @return 1 : Utilisateur interne<br>
     *         0 : Utilisateur externe
```

```
125     */
126    public boolean isInternal() {
127        return user_internal;
128    }
129
130    /**
131     * Sauvegarde le fait que l'utilisateur soit un interne
132     *
133     * @param user_internal
134     *            1 : Utilisateur interne<br>
135     *            0: Utilisateur externe
136     */
137    public void setInternal(boolean user_internal) {
138        this.user_internal = user_internal;
139    }
140
141    /**
142     * Retourne l'état du compte de l'utilisateur
143     *
144     * @return 1 : Compte validé<br>
145     *         0 : Compte non-validé
146     */
147    public boolean getUser_validation() {
148        return user_validation;
149    }
150
151    /**
152     * Sauvegarde le fait que l'utilisateur ait validé son compte ou non
153     *
154     * @param user_validation
155     *            1 : Compte validé<br>
156     *            0 : Compte non-validé
157     */
158    public void setUser_validation(boolean user_validation) {
159        this.user_validation = user_validation;
160    }
161
162    /**
163     * Retourne la date de validation du compte utilisateur
164     *
165     * @return Date de validation
166     */
167    public String getUser_validation_date() {
168        return user_validation_date;
169    }
170
171    /**
172     * Stocke la date de validation du compte utilisateur
173     *
174     * @param user_validation_date
175     *            Date de validation
176     */
177    public void setUser_validation_date(String user_validation_date) {
178        this.user_validation_date = user_validation_date;
179    }
180
181    /**
182     * Retourne la langue préférée par l'utilisateur<br>
183     * Par défaut, la langue préférée est celle que l'utilisateur utilise dans
184     * son navigateur.
185     *
186     * @return Langue de l'utilisateur
```



```
187     */
188    public String getUser_langue() {
189        return user_langue;
190    }
191
192    /**
193     * Stocke la langue préférée de l'utilisateur
194     *
195     * @param user_langue
196     *            Langue de l'utilisateur
197     */
198    public void setUser_langue(String user_langue) {
199        this.user_langue = user_langue;
200    }
201
202    /**
203     * @return Indique si l'utilisateur est un administrateur
204     */
205    public boolean isUser_admin() {
206        return user_admin;
207    }
208
209    /**
210     * @param Définit
211     *            l'utilisateur comme administrateur ou non
212     */
213    public void setUser_admin(boolean user_admin) {
214        this.user_admin = user_admin;
215    }
216
217}
218
```

PACKAGE

MODULES

- AboutModule
- ConditionsModule
- ContactModule
- DownloadModule
- FolderModule
- GetLinkModule
- ReMailLinkModule
- TransfertModule
- **admin**
  - AdminJournModule
  - AdminUsersModule

Modules de l'application qui forment les différentes pages (groupes de composants)

```java
package modules;

import java.util.regex.Matcher;

/**
 * Page d'affichage des informations concernant l'application.<br>
 * La page inclut un bouton de retour au dernier module actif.<br>
 * Les informations sont contenues dans un bloc et illustrées par le logo de la
 * Caisse de compensation
 *
 * @author Dominique Roduit
 *
 */
public class AboutModule {
    /**
     * Contient toutes les instances qui doivent être accessibles dans toute
     * l'application
     **/
    private static GlobalObjects global = CccvsTransfert.getGlobalMethod();
    /** Instance de la fenêtre principale **/
    private static Window mainWindow = CccvsTransfert.mainWindow;
    /** Layout Global de l'application **/
    private static MainLayout mainLayout = global.getMainLayout();
    /** Bouton d'ajout d'un contact **/
    private static Button btAddFolder;
    /** Table contenant la liste des contacts **/
    private static ContactTable tblFolder;

    /**
     * Chargement du bouton de retour et du titre dans le ruban du corps de la
     * page
     */
    public static HorizontalLayout getBodyRibbonContent() {
        System.out.println(Utilities.getCurrentDate()
                + " -- [Chargement du module] Ruban");

        HorizontalLayout vlRibbon = new HorizontalLayout();
        vlRibbon.setSizeFull();
        vlRibbon.setSpacing(true);

        Button btBack = new Button(Global.i("CAPTION_BACK"),
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        //
  if(Global.URLParameters.indexOf(Global.getParm("URL_PARM_DOWNLOAD"))<0)
                        // {
                        String slctTab = AccordionMenu.getSelectedTab();
                        AccordionMenu.selectTab(slctTab);
                    }
                });
        btBack.setStyleName(Runo.BUTTON_DEFAULT);
        vlRibbon.addComponent(btBack);
        vlRibbon.setComponentAlignment(btBack, Alignment.MIDDLE_LEFT);

        Label lblTitle = new Label(" " + Global.i("CAPTION_ABOUT"));
        lblTitle.setStyleName(Runo.LABEL_H1);
        vlRibbon.addComponent(lblTitle);
        vlRibbon.setComponentAlignment(lblTitle, Alignment.MIDDLE_LEFT);

        return vlRibbon;
    }

```



```java
    /**
     * Insertion du contenu dans le corps de la page.<br>
     * Le contenu comprend le texte, extrait de la base de données ainsi qu'une
     * image illustrative.
     */
    public static VerticalLayout getBodyContent() {
        System.out.println(Utilities.getCurrentDate()
                + " -- [Chargement du module] Body");

        VerticalLayout body = new VerticalLayout();
        body.setSizeFull();

        PanelLight pnlContent = new PanelLight();
        pnlContent.setMargin(true);
        pnlContent.setSpacing(true);
        body.addComponent(pnlContent);

        String imgContent = "<div align=\"center\" style=\"margin-top:8px\"><img
  src=\""
                + Global.PATH_THEME_RESSOURCES_HTML
                + "logoAbout.png\" /></div>";

        Label lblContent = new Label(
                imgContent + "<div class=\"content-area-about\">"
                        + Utilities.parmConverter(Global.i("CONTENT_ABOUT"))
                        + "</div>", Label.CONTENT_XHTML);
        pnlContent.addComponent(lblContent);

        return body;
    }
}

```

```java
package modules;

import main.CccvsTransfert;

/**
 * Page d'affichage des conditions d'utilisation de l'application.<br>
 * La page inclut un bouton de retour au dernier module actif.<br>
 * Les conditions sont contenues dans un bloc et illustrées par une image libre
 * au format PNG.
 *
 * @author Dominique Roduit
 *
 */
public class ConditionsModule {
    /**
     * Contient toutes les instances qui doivent être accessibles dans toute
     * l'application
     **/
    private static GlobalObjects global = CccvsTransfert.getGlobalMethod();
    /** Instance de la fenêtre principale **/
    private static Window mainWindow = CccvsTransfert.mainWindow;
    /** Layout Global de l'application **/
    private static MainLayout mainLayout = global.getMainLayout();
    /** Bouton d'ajout d'un contact **/
    private static Button btAddFolder;
    /** Table contenant la liste des contacts **/
    private static ContactTable tblFolder;

    /**
     * Chargement du bouton de retour et du titre dans le ruban du corps de la
     * page
     */
    public static HorizontalLayout getBodyRibbonContent() {
        System.out.println(Utilities.getCurrentDate()
                + " -- [Chargement du module] Ruban");

        HorizontalLayout vlRibbon = new HorizontalLayout();
        vlRibbon.setSizeFull();
        vlRibbon.setSpacing(true);

        Button btBack = new Button(Global.i("CAPTION_BACK"),
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        String slctTab = AccordionMenu.getSelectedTab();
                        AccordionMenu.selectTab(slctTab);
                    }
                });
        btBack.setStyleName(Runo.BUTTON_DEFAULT);
        vlRibbon.addComponent(btBack);
        vlRibbon.setComponentAlignment(btBack, Alignment.MIDDLE_LEFT);

        Label lblTitle = new Label(Global.i("CAPTION_USAGE_CONDITIONS"));
        lblTitle.setStyleName(Runo.LABEL_H1);
        vlRibbon.addComponent(lblTitle);
        vlRibbon.setComponentAlignment(lblTitle, Alignment.MIDDLE_LEFT);

        Button btUserManual = new Button(Global.i("CAPTION_USER_MANUAL"),
                new Button.ClickListener() {
                    @Override
                    public void buttonClick(ClickEvent event) {
                        Window winUserManual = new Window(
                                Global.i("CAPTION_USER_MANUAL"));
```



```java
                        winUserManual.setWidth("880px");
                        winUserManual.setHeight("580px");
                        winUserManual.center();
                        winUserManual.setStyleName(Reindeer.WINDOW_BLACK);
                        winUserManual.setResizable(false);
                        winUserManual.setModal(true);
                        CccvsTransfert.mainWindow.addWindow(winUserManual);

                        Embedded pdf = new Embedded(null, new ThemeResource(
                                Global.PATH_THEME_RESSOURCES
                                        + "user-manual.pdf"));
                        pdf.setMimeType("application/pdf");
                        pdf.setType(Embedded.TYPE_BROWSER);
                        pdf.setHeight("520px");
                        pdf.setWidth("845px");
                        winUserManual.addComponent(pdf);

                    }
                });
        btUserManual.setIcon(new ThemeResource(Global.PATH_THEME_RESSOURCES
                + "usermanual.png"));
        btUserManual.setStyleName("ribbon-right-alignment");
        vlRibbon.addComponent(btUserManual);
        vlRibbon.setComponentAlignment(btUserManual, Alignment.MIDDLE_RIGHT);

        return vlRibbon;
    }

    /**
     * Insertion du contenu dans le corps de la page.<br>
     * Le contenu comprend le texte, extrait de la base de données ainsi qu'une
     * image illustrative.
     */
    public static VerticalLayout getBodyContent() {
        System.out.println(Utilities.getCurrentDate()
                + " -- [Chargement du module] Body");

        VerticalLayout body = new VerticalLayout();
        body.setSizeFull();

        PanelLight pnlContent = new PanelLight();
        pnlContent.setMargin(true);
        pnlContent.setSpacing(true);
        body.addComponent(pnlContent);

        String imgContent = "<div align=\"center\" style=\"margin-top:8px\"><img
    src=\""
                + Global.PATH_THEME_RESSOURCES_HTML + "book.png\" /></div>";

        Label lblContent = new Label(imgContent
                + "<div class=\"content-area-about\">"
                + Utilities.parmConverter(Global.i("CONTENT_USAGE_CONDITIONS"))
                + "</div>", Label.CONTENT_XHTML);
        pnlContent.addComponent(lblContent);

        return body;
    }
}
```

```java
package modules;

import main.CccvsTransfert;

/**
 * Page de gestion des contacts.<br>
 * L'utilisateur peut ajouter/éditer/supprimer des contacts. Les actions
 * effectuées sur cette page sont répercutées sur le tableau du menu affichant
 * le même résultat
 *
 * @author Dominique Roduit
 *
 */
public class ContactModule {
    /**
     * Contient toutes les instances qui doivent être accessibles dans toute
     * l'application
     **/
    private static GlobalObjects global = CccvsTransfert.getGlobalMethod();
    /** Instance de la fenêtre principale **/
    private static Window mainWindow = CccvsTransfert.mainWindow;
    /** Layout Global de l'application **/
    private static MainLayout mainLayout = global.getMainLayout();
    /** Bouton d'ajout d'un contact **/
    private static Button btAddContact;
    /** Table contenant la liste des contacts **/
    private static ContactTable tableContact;
    /** Contient la fenêtre d'ajout/édition d'un contact **/
    private static Window winContact;

    /**
     * Chargement des boutons dans le ruban du corps de page
     */
    public static HorizontalLayout getBodyRibbonContent() {
        System.out.println(Utilities.getCurrentDate()
                + " -- [Chargement du module] Ruban");

        HorizontalLayout hRibbon = new HorizontalLayout();
        hRibbon.setSizeFull();

        btAddContact = new Button(Global.i("CAPTION_CONTACT_ADD"));
        btAddContact.setIcon(new ThemeResource(Global.PATH_THEME_RESSOURCES
                + "add-contact.png"));
        btAddContact.focus();
        global.setBtAddContact(btAddContact);

        btAddContact.addListener(new Button.ClickListener() {
            @Override
            public void buttonClick(ClickEvent event) {
                System.out.println(winContact);

                Window winContact = new Window(Global.i("CAPTION_CONTACT_ADD"));
                winContact.setModal(true);
                winContact.setWidth("440px");
                winContact.setHeight("238px");
                winContact.setResizable(false);
                CccvsTransfert.mainWindow.addWindow(winContact);
                global.setWindowContact(winContact);

                ContactForm cf;
                // Si le bouton ne possède aucun style
                if (btAddContact.getStyleName().equals("")) { // On ajoute un
```



```java
                                                              // nouveau
                                                              // contact
                    cf = new ContactForm();
                } else { // Sinon, on met à jour le contact sélectionné
                    cf = new ContactForm(btAddContact.getStyleName()
                            .replace("[", "").replace("]", ""));
                    btAddContact.removeStyleName(btAddContact.getStyleName());
                }
                winContact.addComponent(cf);
            }
        });
        hRibbon.addComponent(btAddContact);
        hRibbon.setComponentAlignment(btAddContact, Alignment.MIDDLE_LEFT);

        int nbContact = tbl_contacts.getNumberOfContact();
        if (nbContact < 1) {
            AbsoluteLayout absFinger = new AbsoluteLayout();
            absFinger.setWidth("46px");
            absFinger.setHeight("32px");
            absFinger.setStyleName("finger");
            hRibbon.addComponent(absFinger);
            hRibbon.setComponentAlignment(absFinger, Alignment.MIDDLE_LEFT);
        }

        return hRibbon;
    }

    /**
     * Insertion du contenu dans le corps de la page.<br>
     * Le contenu comprend un tableau répertoriant les contact que l'utilisateur
     * à ajouté.<br>
     * Le tableau implémente un menu contextuel qui permet la suppression et
     * l'édition d'un contact
     */
    public static ContactTable getBodyContent() {
        System.out.println(Utilities.getCurrentDate()
                + " -- [Chargement du module] Body");

        tableContact = new ContactTable();
        global.setTableContact(tableContact);
        return tableContact;
    }
}
```

```java
package modules;

import global.Global;

/**
 * Page de téléchargement des fichiers d'un dossier dans lequel un ou plusieurs
 * contacts ont été invités.
 *
 * @author Dominique Roduit
 * @since 2013-03-08
 *
 */
public class DownloadModule {
    /**
     * Contient toutes les instances qui doivent être accessibles dans toute
     * l'application
     **/
    private GlobalObjects global = CccvsTransfert.getGlobalMethod();
    /** Instance de la fenêtre principale **/
    private Window mainWindow = CccvsTransfert.mainWindow;
    /** Layout Global de l'application **/
    private MainLayout mainLayout = global.getMainLayout();
    /** Bouton d'ajout d'un contact **/
    private Button btDownloadAll;
    /** Table contenant la liste des contacts **/
    private Table tblFiles;
    /** Clé primaire du dossier **/
    private int PKNoFolder = 0;
    /** Clé primaire du contact qui accède au dossier **/
    private int PKNoContact = 0;
    /** Adresse e-mail du contact qui accède au dossier **/
    private String mail = "";
    /** Label contenant le nom du dossier **/
    private Label lblFolderName;
    /** Modèle contenant les informations du dossier **/
    private Folder folder;
    /** Modèle contenant les informations sur l'auteur du dossier **/
    private User auteur;
    /** Container pour les actions journalisées du tableau **/
    private BeanItemContainer<ActionJournalFold> containerActions = new
BeanItemContainer<ActionJournalFold>(
            ActionJournalFold.class);
    /** Table du footer contenant les actions journalisées **/
    private Table tblActions;
    /** Container contenant la liste des fichiers du dossier **/
    private BeanItemContainer<Files> container = new BeanItemContainer<Files>(
            Files.class);
    /** Action "Télécharger" du menu contextuel **/
    private Action ACTION;
    /** Nombre de consultation du dossier par le contact **/
    private int nbConsultation = 0;

    /**
     * Création d'une page de téléchargement des fichiers d'un dossier
     * spécifique
     *
     * @param PKNoFolder
     *            Clé primaire du dossier consulté
     * @param PKNoContact
     *            Clé primaire du contact invité au dossier
     * @param mail
     *            Adresse e-mail du contact invité au dossier
```



```java
     */
    public DownloadModule(int PKNoFolder, int PKNoContact, String mail) {
        this.PKNoFolder = PKNoFolder;
        this.PKNoContact = PKNoContact;
        this.mail = mail;

        ACTION = new Action(Global.i("CAPTION_DOWNLOAD"), new ThemeResource(
                Global.PATH_THEME_RESSOURCES + "down.png"));

        // Récupération et stockage des informations sur le dossier
        ResultSet info = Sql
                .query(tbl_folders.getSelectFolderInfos(PKNoFolder));
        folder = new Folder();
        folder.setPKNoFolder(PKNoFolder);
        try {
            if (info.first()) {
                folder.setName(info.getString("folder_name"));
                folder.setCreation_date(info.getString("folder_creation_date"));
                folder.setExpiration_date(info.getString("folder_expiration"));
                folder.setArchive(info.getBoolean("folder_archive"));
                folder.setFKNoUser(info.getInt("FKNoUser"));
                folder.setDescription(info.getString("folder_description"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        // Récupération et stockage des informations sur l'auteur du dossier
        ResultSet iAut = Sql.query(tbl_users.getSelectUserInfos(folder
                .getFKNoUser()));
        auteur = new User();
        auteur.setPKNoUser(folder.getFKNoUser());
        try {
            if (iAut.first()) {
                auteur.setUser_mail(iAut.getString("user_mail"));
                auteur.setInternal(iAut.getBoolean("user_internal"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        // Journalisation de la consultation du dossier
        Sql.exec(tbl_journal_fold.getInsertQuery("consult", PKNoContact,
                PKNoFolder, 0));

        // Chargement du contenu du footer
        tblActions = new Table();
        tblActions.setSizeFull();
        tblActions.addContainerProperty("action_img", Embedded.class, null, "",
                null, Table.ALIGN_CENTER);
        tblActions.addContainerProperty("action_type", String.class, null,
                Global.i("CAPTION_ACTION"), null, null);
        tblActions.addContainerProperty("action_description", String.class,
                null, Global.i("CAPTION_DESCRIPTION"), null, null);
        tblActions.addContainerProperty("action_date", String.class, null,
                Global.i("CAPTION_DATE"), null, null);
        tblActions.setColumnWidth("action_img", 30);
        tblActions.setColumnWidth("action_type", 150);
        mainLayout.getFooter().addComponent(tblActions);

        ResultSet actionsJournal = Sql.query(tbl_journal_fold
                .getSelectAllForContactAndFolder(PKNoContact, PKNoFolder));
```

```
174         try {
175             while (actionsJournal.next()) {
176                 if (actionsJournal.getString("joufo_action").equals("download")) {
177                     addActionDownloadItem(
178                             (actionsJournal.getString("FKNoFile") != null) ?
    actionsJournal.getString("file_rename")
179                                     : Global.i("CAPTION_ARCHIVE_OF_FOLDER"),
180                             actionsJournal.getString("joufo_date"),
181                             (actionsJournal.getString("FKNoFile") != null) ? true
182                                     : false);
183                 } else {
184                     addActionConsultItem(actionsJournal.getString("joufo_date"));
185                 }
186             }
187         } catch (SQLException e) {
188             e.printStackTrace();
189         }
190     }
191 
192     /**
193      * Chargement des boutons dans le ruban du corps de la page
194      */
195     public HorizontalLayout getBodyRibbonContent() {
196         HorizontalLayout vlRibbon = new HorizontalLayout();
197         vlRibbon.setSizeFull();
198         vlRibbon.setWidth("96%");
199         mainLayout.getBodyRibbon().addComponent(vlRibbon);
200 
201         lblFolderName = new Label(folder.getName());
202         lblFolderName.setStyleName(Runo.LABEL_H1);
203         vlRibbon.addComponent(lblFolderName);
204         vlRibbon.setComponentAlignment(lblFolderName, Alignment.MIDDLE_LEFT);
205 
206         Button btDownload = new Button(Global.i("CAPTION_DOWNLOAD_ALL"),
207                 new Button.ClickListener() {
208                     public void buttonClick(ClickEvent event) {
209                         String archiveName = Utilities.getFolderArchiveName(
210                                 folder.getName(), folder.getCreation_date());
211 
212                         System.out.println(archiveName);
213 
214                         File internFile = new File(Global.UPLOAD_DIR
215                                 + archiveName);
216                         if (internFile.exists()) {
217                             CccvsTransfert.mainWindow.open(
218                                     new ExternalResource(Utilities
219                                             .getURLForFile(archiveName)),
220                                     "_self");
221                             addActionDownloadItem(
222                                     Global.i("CAPTION_ARCHIVE_OF_FOLDER"),
223                                     false);
224                             Sql.exec(tbl_journal_fold.getInsertQuery(
225                                     "download", PKNoContact, PKNoFolder, 0));
226                         } else {
227                             mainWindow.showNotification(
228                                     Global.i("CAPTION_INEXISTING_FILE"),
229                                     Notification.TYPE_WARNING_MESSAGE);
230                         }
231 
232                     }
233                 });
234         btDownload.setIcon(new ThemeResource(Global.PATH_THEME_RESSOURCES
```



```
235                 + "download.png"));
236         btDownload.setStyleName("ribbon-right-alignment");
237         vlRibbon.addComponent(btDownload);
238         vlRibbon.setComponentAlignment(btDownload, Alignment.MIDDLE_RIGHT);
239 
240         return vlRibbon;
241     }
242 
243     /**
244      * Insertion du contenu dans le corps de la page.<br>
245      * Le contenu comprend un tableau listant les fichiers du dossier ainsi que
246      * les informations y relatives.
247      */
248     public Table getBodyContent() {
249         System.out.println(Utilities.getCurrentDate()
250                 + " -- [Chargement du module] Body");
251 
252         ArrayList<Files> fileList = getFileList();
253         container.removeAllItems();
254         container.addAll(fileList);
255 
256         // Table contenant les fichiers du dossier
257         tblFiles = new Table();
258         tblFiles.setSizeFull();
259         tblFiles.setSelectable(true);
260         tblFiles.setContainerDataSource(container);
261         tblFiles.setVisibleColumns(new String[] { "file_rename",
262                 "file_size_formatted", "file_extension", "file_description" });
263         tblFiles.setColumnHeaders(new String[] { Global.i("CAPTION_FILENAME"),
264                 Global.i("CAPTION_SIZE"), Global.i("CAPTION_TYPE"),
265                 Global.i("CAPTION_DESCRIPTION") });
266         tblFiles.setColumnCollapsingAllowed(true);
267         tblFiles.setColumnReorderingAllowed(true);
268         tblFiles.setNullSelectionAllowed(false);
269         tblFiles.addListener(new ItemClickListener() {
270             public void itemClick(ItemClickEvent event) {
271                 if (event.getButton() == ItemClickEvent.BUTTON_LEFT) {
272                     Files slctFile = ((Files) event.getItemId());
273                     System.out.println(slctFile.getFile_name() + " - "
274                             + slctFile.getFile_size_formatted());
275                 }
276             }
277         });
278         tblFiles.setColumnWidth("file_size_formatted", 100);
279         tblFiles.setColumnWidth("file_extension", 60);
280 
281         // Ajout de la colonne des boutons de téléchargement
282         tblFiles.addGeneratedColumn("download", new Table.ColumnGenerator() {
283             public Component generateCell(Table source, Object itemId,
284                     Object columnId) {
285                 final Files file = (Files) itemId;
286 
287                 Button btDown = new Button(Global.i("CAPTION_DOWNLOAD"),
288                         new Button.ClickListener() {
289                             public void buttonClick(ClickEvent event) {
290                                 downloadFile(file);
291                             }
292                         });
293 
294                 btDown.setStyleName(Runo.BUTTON_SMALL);
295                 return btDown;
296             }
```

```
297         });
298         tblFiles.setColumnHeader("download", Global.i("CAPTION_ACTION"));
299         tblFiles.setColumnWidth("download", 90);
300         tblFiles.setColumnAlignment("download", Table.ALIGN_CENTER);
301
302         // Ajout de la colonne des icones
303         tblFiles.addGeneratedColumn("icone", new Table.ColumnGenerator() {
304             public Object generateCell(Table source, Object itemId,
305                     Object columnId) {
306                 Files file = (Files) itemId;
307                 return Utilities.getImgFromExtension(file.getFile_extension());
308             }
309         });
310         tblFiles.setColumnHeader("icone", "");
311         tblFiles.setColumnWidth("icone", 28);
312         tblFiles.setColumnAlignment("icone", Table.ALIGN_CENTER);
313         tblFiles.setColumnCollapsible("icone", false);
314
315         // Réorganisation des colonnes
316         tblFiles.setVisibleColumns(new String[] { "icone", "file_rename",
317                 "file_size_formatted", "file_extension", "file_description",
318                 "download" });
319
320         tblFiles.addActionHandler(new Action.Handler() {
321             public void handleAction(Action action, Object sender, Object target) {
322                 if (action == ACTION) {
323                     Files file = (Files) target;
324                     downloadFile(file);
325                 }
326             }
327
328             public Action[] getActions(Object target, Object sender) {
329                 if (target != null) {
330                     return new Action[] { ACTION };
331                 } else {
332                     return null;
333                 }
334             }
335         });
336         tblFiles.addListener(new ItemClickListener() {
337             public void itemClick(ItemClickEvent event) {
338                 if (event.getButton() == ItemClickEvent.BUTTON_RIGHT) {
339                     tblFiles.setValue(null); // Dé-sélectionne tout
340                     tblFiles.select(event.getItemId()); // Sélectionne la ligne
341                                                         // en cours
342                 }
343             }
344         });
345
346         VerticalLayout vlInfo = new VerticalLayout();
347         vlInfo.setWidth(Global.getParm("LAYOUT_FOLDER_INFO_WIDTH"));
348         vlInfo.setHeight("100%");
349         vlInfo.setStyleName("info-folder");
350
351         // Récupération des destinataires qui ont été ajoutés au dossier
352         ArrayList<Contact> recipient = new ArrayList<Contact>();
353         ResultSet dest = Sql.query(tbl_recipients
354                 .getSelectRecipientsFromFolder(PKNoFolder));
355         try {
356             while (dest.next()) {
357                 recipient.add(new Contact(dest.getInt("PKNoContact"), dest
358                         .getString("contact_name"), dest
```



```
359                         .getString("contact_mail"), dest
360                         .getBoolean("contact_internal")));
361             }
362         } catch (SQLException e) {
363             e.printStackTrace();
364         }
365
366         String restant = "";
367         int jRestant = tbl_folders
368                 .getRemainingDaysFolderIsAvailable(PKNoFolder);
369         int hRestant = tbl_folders
370                 .getRemainingHoursFolderIsAvailable(PKNoFolder);
371         restant = (jRestant > 0) ? "<span style=\"color:green\">" + jRestant
372                 + " " + Global.i("CAPTION_DAYS").toLowerCase() + "</span>"
373                 : "<span style=\"color:orange\">"
374                         + ((Integer.parseInt(Utilities.formatSQLDate(
375                                 folder.getExpiration_date(), "mm")) > 30) ?
    (hRestant + 1)
376                                 : hRestant) + " "
377                         + Global.i("CAPTION_HOURS").toLowerCase() + "</span>";
378
379         if (jRestant == 0 && hRestant <= 0) {
380             restant = "<span style=\"color:red\">"
381                     + Global.i("CAPTION_ARCHIVAGE_AT")
382                     + " "
383                     + Utilities
384                             .zeroFill(Integer.parseInt(Utilities.formatSQLDate(
385                                     folder.getCreation_date(), "HH")) + 1)
386                     + ":00</span>";
387         }
388
389         String contentLabel = "<div class=\"infos\">" + "<h2>"
390                 + Utilities.htmlentities(folder.getName())
391                 + "</h2>"
392                 + "<div class=\"filesize\">"
393                 + "<div style=\"float:left; width:50%; border-right: 1px solid
    #ddd\">"
394                 + tbl_files.getFileNumberFromFolder(PKNoFolder)
395                 + " "
396                 + Global.i("CAPTION_FILES").toLowerCase()
397                 + "</div>"
398                 + "<div style=\"float:left; width:49%;\">"
399                 + Utilities.formatSize(tbl_files
400                         .getFileSizeFromFolder(PKNoFolder))
401                 + "</div>"
402                 + "<div style=\"clear:both\"></div>"
403                 + "</div>"
404                 + "<div class=\"dates\">"
405                 + "<table>"
406                 + "<tr><th>"
407                 + Global.i("CAPTION_CREATION")
408                 + "</th> <th>"
409                 + Global.i("CAPTION_EXPIRATION")
410                 + "</th> <th>"
411                 + Global.i("CAPTION_AVAILABILITY")
412                 + "</th></tr>"
413                 + "<tr><td>"
414                 + Utilities.formatSQLDateWtText(folder.getCreation_date(),
415                         "dd MMMM YY")
416                 + "</td> <td>"
417                 + Utilities.formatSQLDateWtText(folder.getExpiration_date(),
418                         "dd MMMM YY") + "</td> <td>" + restant + "</td>"
```

```
419                 + "</table>" + "</div>";
420
421         if (folder.getDescription() != null) {
422             if (folder.getDescription().length() > 0) {
423                 contentLabel += "<div class=\"description\">"
424                         + Utilities.htmlentities(folder.getDescription())
425                         + "</div>";
426             }
427         }
428
429         contentLabel += "<p class=\"title\">" + Global.i("CAPTION_AUTHOR")
430                 + "</p>" + "<ul><li>" + auteur.getUser_mail() + "</li></ul>";
431
432         if (recipient.size() > 0) {
433             contentLabel += "<p class=\"title\">"
434                     + Global.i("CAPTION_RECIPIENTS") + " (" + recipient.size()
435                     + ")</p>" + "<ul>";
436             for (Contact contact : recipient) {
437                 String liDest = "<li>" + contact.getName() + "</li>";
438                 if (contact.getPK() == PKNoContact) {
439                     liDest = "<li style=\"font-weight:bold\">"
440                             + contact.getName() + " (Moi)</li>";
441                 }
442                 contentLabel += liDest;
443             }
444             contentLabel += "</ul>";
445         } else {
446             contentLabel += "<br><div align=\"center\"><b>"
447                     + Global.i("CAPTION_NO_RECIPIENT") + "</b></div>";
448         }
449
450         contentLabel += "</div>";
451
452         Label lblInfos = new Label(contentLabel, Label.CONTENT_XHTML);
453         vlInfo.addComponent(lblInfos);
454
455         mainLayout.getMenu().addComponent(vlInfo);
456
457         return tblFiles;
458     }
459
460     /**
461      * Lance le téléchargement d'un fichier et journalise l'opération
462      *
463      * @param file
464      *            Informations sur le fichier
465      */
466     private void downloadFile(Files file) {
467         File fileToDownload = new File(Global.UPLOAD_DIR + file.getFile_name());
468
469         if (fileToDownload.exists()) {
470             CccvsTransfert.mainWindow.open(
471                     new ExternalResource(Utilities.getURLForFile(file
472                             .getFile_name())), "_self");
473             // Journalisation
474             Sql.exec(tbl_journal_fold.getInsertQuery("download", PKNoContact,
475                     PKNoFolder, file.getPKNoFile()));
476             addActionDownloadItem(file.getFile_rename(), true);
477
478         } else {
479             mainWindow.showNotification(Global.i("CAPTION_INEXISTING_FILE"),
480                     Notification.TYPE_WARNING_MESSAGE);
```



```
481         }
482
483     }
484
485     /**
486      * Ajoute une action de journalisation pour un téléchargement dans la table
487      *
488      * @param filename
489      *            Nom du fichier téléchargé
490      */
491     private void addActionDownloadItem(String filename, String file_date,
492             boolean isFile) {
493         Embedded imgIllustr = new Embedded(null, new ThemeResource(
494                 (isFile) ? Global.IMG_DOWNLOAD : Global.IMG_ARCHIVE_DOWNLOAD));
495         tblActions.addItem(
496                 new Object[] {
497                         imgIllustr,
498                         Global.i("CAPTION_DOWNLOADING"),
499                         filename,
500                         Utilities.formatSQLDate(file_date,
501                                 "dd MMMM YY HH:mm:ss") }, tblActions.size());
502         tblActions.setCurrentPageFirstItemIndex(tblActions.size());
503     }
504
505     /**
506      * Ajoute une action de journalisation pour un téléchargement dans la table
507      *
508      * @param filename
509      *            Nom du fichier téléchargé
510      */
511     private void addActionDownloadItem(String filename, boolean isFile) {
512         addActionDownloadItem(filename, Utilities.getCurrentDate(), isFile);
513     }
514
515     /**
516      * Ajoute une action de journalisation pour une consultation dans la table
517      */
518     private void addActionConsultItem(String date) {
519         nbConsultation++;
520         Embedded imgIllustr = new Embedded(null, new ThemeResource(
521                 Global.IMG_SEARCH));
522         tblActions
523                 .addItem(
524                         new Object[] {
525                                 imgIllustr,
526                                 Global.i("CAPTION_CONSULTATION"),
527                                 Global.i("CAPTION_CONSULTATION") + " "
528                                         + nbConsultation,
529                                 Utilities.formatSQLDate(date,
530                                         "dd MMMM YY HH:mm:ss") },
531                         tblActions.size());
532         tblActions.setCurrentPageFirstItemIndex(tblActions.size());
533     }
534
535     /**
536      * Ajoute une action de journalisation pour une consultation dans la table
537      */
538     private void addActionConsultItem() {
539         addActionConsultItem(Utilities.getCurrentDate());
540     }
541
542     /**
```

```
543     * Retourne la liste des fichiers contenus dans le dossier
544     *
545     * @return Liste des fichiers du dossier
546     */
547    private ArrayList<Files> getFileList() {
548        ArrayList<Files> files = new ArrayList<Files>();
549        ResultSet data = Sql.query(tbl_files
550                .getSelectAllFilesFromFolder(PKNoFolder));
551        try {
552            while (data.next()) {
553                files.add(new Files(data.getInt("PKNoFile"), data
554                        .getInt("FKNoFolder"), data.getString("file_name"),
555                        data.getString("file_rename"), data
556                                .getLong("file_size"), data
557                                .getString("file_extension"), data
558                                .getString("file_description"), Utilities
559                                .formatSize(data.getLong("file_size"))));
560            }
561        } catch (SQLException e) {
562            e.printStackTrace();
563        }
564        return files;
565    }
566}
```



```
1 package modules;
2
3 import java.io.File;
48
49/**
50 * Page d'affichage des fichiers et informations d'un dossier.<br>
51 * Le ruban n'est pas rechargé par cette class, le bouton d'ajout d'un dossier
52 * est donc conservé.<br>
53 * Un panneau affiche les informations du dossier<br>
54 * <ul>
55 * <li>Création</li>
56 * <li>Expiration</li>
57 * <li>Disponibilité</li>
58 * <li>Destinataires</li>
59 * <li>Statut (Archivé ou non)</li>
60 * <li>...</li>
61 * </ul>
62 * <br>
63 * <br>
64 * Les types de fichiers sont illustrés par une icône d'extension.
65 *
66 * @author Dominique Roduit
67 *
68 */
69public class FolderModule {
70    /**
71     * Contient toutes les instances qui doivent être accessibles dans toute
72     * l'application
73     **/
74    private static GlobalObjects global = CccvsTransfert.getGlobalMethod();
75    /** Instance de la fenêtre principale **/
76    private static Window mainWindow = CccvsTransfert.mainWindow;
77    /** Layout Global de l'application **/
78    private static MainLayout mainLayout = global.getMainLayout();
79    /** Bouton d'ajout d'un contact **/
80    private static Button btAddFolder;
81    /** Table contenant la liste des contacts **/
82    private static ContactTable tblFolder;
83    /** Contient la fenêtre de création d'un dossier **/
84    private static Window winFolder;
85    /** Contient la clé primaire du dossier à afficher **/
86    private static int FKNoFolder = 0;
87    /** Container contenant la liste des fichiers du dossier **/
88    private static BeanItemContainer<Files> container = new
  BeanItemContainer<Files>(
89            Files.class);
90    /** Table contenant les fichier contenus dans le dossier **/
91    private static Table tblFiles;
92    /** Action "Télécharger" du menu contextuel **/
93    private static Action ACTION_DOWNLOAD;
94    /** Action pour la récupération du lien de téléchargement **/
95    protected static Action ACTION_GET_LINK;
96
97    /**
98     * Chargement des boutons dans le ruban du corps de page
99     */
100   public static Button getBodyRibbonContent() {
101       return null;
102   }
103
104   /**
105    * Insertion du contenu dans le corps de page.<br>
```

```
106     * Le contenu comprend le tableau répertoriant les fichiers ainsi que les
107     * informations relatives au dossier.
108     */
109    public static HorizontalLayout getBodyContent(int PKNoFolder) {
110        System.out.println(Utilities.getCurrentDate()
111                + " -- [Chargement du module] Body");
112
113        ACTION_DOWNLOAD = new Action(Global.i("CAPTION_DOWNLOAD"),
114                new ThemeResource(Global.PATH_THEME_RESSOURCES + "down.png"));
115        ACTION_GET_LINK = new Action(Global.i("CAPTION_GET_DOWNLOAD_LINK"),
116                new ThemeResource(Global.PATH_THEME_RESSOURCES + "link.png"));
117
118        FKNoFolder = PKNoFolder;
119
120        HorizontalLayout body = new HorizontalLayout();
121        body.setSizeFull();
122
123        GridLayout grdLayout = new GridLayout(2, 1);
124        grdLayout.setColumnExpandRatio(0, 100);
125        grdLayout.setSizeFull();
126        body.addComponent(grdLayout);
127
128        ArrayList<Files> fileList = getFileList();
129        container.removeAllItems();
130        container.addAll(fileList);
131
132        tblFiles = new Table();
133        tblFiles.setSizeFull();
134        tblFiles.setSelectable(true);
135        tblFiles.setContainerDataSource(container);
136        tblFiles.setVisibleColumns(new String[] { "file_rename",
137                "file_size_formatted", "file_extension", "file_description" });
138        tblFiles.setColumnHeaders(new String[] { Global.i("CAPTION_FILENAME"),
139                Global.i("CAPTION_SIZE"), Global.i("CAPTION_TYPE"),
140                Global.i("CAPTION_DESCRIPTION") });
141        tblFiles.setColumnCollapsingAllowed(true);
142        tblFiles.setColumnReorderingAllowed(true);
143        tblFiles.setNullSelectionAllowed(false);
144        tblFiles.setColumnCollapsed("file_description", true);
145        // Clique d'une ligne du tableau
146        tblFiles.addListener(new ItemClickListener() {
147            public void itemClick(ItemClickEvent event) {
148                if (event.getButton() == ItemClickEvent.BUTTON_LEFT) {
149                    Files slctFile = ((Files) event.getItemId());
150                    System.out.println(slctFile.getPKNoFile() + " : "
151                            + slctFile.getFile_name() + " - "
152                            + slctFile.getFile_size_formatted());
153                }
154            }
155        });
156        tblFiles.setColumnWidth("file_size_formatted", 90);
157        tblFiles.setColumnWidth("file_extension", 70);
158
159        // Ajout de la colonne des icones
160        tblFiles.addGeneratedColumn("icon", new Table.ColumnGenerator() {
161            public Object generateCell(Table source, Object itemId,
162                    Object columnId) {
163                Files file = (Files) itemId;
164                return Utilities.getImgFromExtension(file.getFile_extension());
165            }
166        });
167        tblFiles.setColumnHeader("icon", "");
```



```
168        tblFiles.setColumnWidth("icon", 28);
169        tblFiles.setColumnAlignment("icon", Table.ALIGN_CENTER);
170        tblFiles.setColumnCollapsible("icon", false);
171
172        // Réorganisation des colonnes
173        tblFiles.setVisibleColumns(new String[] { "icon", "file_rename",
174                "file_size_formatted", "file_extension", "file_description" });
175        grdLayout.addComponent(tblFiles, 0, 0);
176
177        // Menu contextuel lors du clique droit sur un fichier
178        tblFiles.addActionHandler(new Action.Handler() {
179            public void handleAction(Action action, Object sender, Object target) {
180                if (action == ACTION_DOWNLOAD) {
181                    Files file = (Files) target;
182                    File fileToDownload = new File(Global.UPLOAD_DIR
183                            + file.getFile_name());
184
185                    if (fileToDownload.exists()) {
186                        CccvsTransfert.mainWindow.open(new ExternalResource(
187                                Utilities.getURLForFile(file.getFile_name())),
188                                "_self");
189                    } else {
190                        CccvsTransfert.mainWindow.showNotification(
191                                Global.i("CAPTION_INEXISTING_FILE"),
192                                Notification.TYPE_WARNING_MESSAGE);
193                    }
194                }
195
196                if (action == ACTION_GET_LINK) {
197                    Files file = (Files) target;
198                    global.openWindowGetLink(file.getFile_name());
199                }
200            }
201
202            public Action[] getActions(Object target, Object sender) {
203                if (target != null) {
204                    return new Action[] { ACTION_DOWNLOAD, ACTION_GET_LINK };
205                } else {
206                    return null;
207                }
208            }
209        });
210        tblFiles.addListener(new ItemClickListener() {
211            public void itemClick(ItemClickEvent event) {
212                if (event.getButton() == ItemClickEvent.BUTTON_RIGHT) {
213                    tblFiles.setValue(null); // Dé-sélectionne tout
214                    tblFiles.select(event.getItemId()); // Sélectionne la ligne
215                                                        // en cours
216                }
217            }
218        });
219
220        VerticalLayout vlInfo = new VerticalLayout();
221        vlInfo.setWidth(Global.getParm("LAYOUT_FOLDER_INFO_WIDTH"));
222        vlInfo.setHeight("100%");
223        vlInfo.setStyleName("info-folder");
224        // vlInfo.setMargin(true);
225        grdLayout.addComponent(vlInfo, 1, 0);
226
227        GridLayout grdInfos = new GridLayout(1, 3);
228        grdInfos.setSizeFull();
229        vlInfo.addComponent(grdInfos);
```

```
230
231         // Récupération des informations sur le dossier
232         ResultSet info = Sql
233                 .query(tbl_folders.getSelectFolderInfos(PKNoFolder));
234
235         Folder folder = null;
236         try {
237             if (info.first()) {
238                 folder = new Folder(info.getInt("PKNoFolder"),
239                         info.getInt("FKNoUser"), info.getString("folder_name"),
240                         info.getString("folder_description"),
241                         info.getString("folder_creation_date"),
242                         info.getString("folder_expiration"),
243                         info.getBoolean("folder_archive"));
244             }
245         } catch (SQLException e) {
246             e.printStackTrace();
247         }
248
249         String contentLabel = "<div class=\"infos\">"
250                 + "<h2>"
251                 + Utilities.htmlentities(folder.getName())
252                 + "</h2>"
253                 + "<div class=\"filesize\">"
254                 + "<div style=\"float:left; width:50%; border-right: 1px solid
    #ddd\">"
255                 + tbl_files.getFileNumberFromFolder(PKNoFolder)
256                 + " "
257                 + Global.i("CAPTION_FILES").toLowerCase()
258                 + "</div>"
259                 + "<div style=\"float:left; width:49%;\">"
260                 + Utilities.formatSize(tbl_files
261                         .getFileSizeFromFolder(PKNoFolder)) + "</div>"
262                 + "<div style=\"clear:both\"></div>" + "</div>"
263                 + "<div class=\"dates\">";
264
265         if (folder.getArchive()) { // Dossier archivé
266             contentLabel += "<table>"
267                     + "<tr><th>"
268                     + Global.i("CAPTION_CREATION")
269                     + "</th> <th>"
270                     + Global.i("CAPTION_EXPIRED")
271                     + "</th></tr>"
272                     + "<tr><td>"
273                     + Utilities.formatSQLDateWtText(folder.getCreation_date(),
274                             "dd MMMM YY")
275                     + "</td> <td>"
276                     + Utilities.formatSQLDateWtText(
277                             folder.getExpiration_date(), "dd MMMM YY")
278                     + "</td></tr>" + "</table>";
279         } else { // Dossier disponible
280             String restant = "";
281             int jRestant = tbl_folders
282                     .getRemainingDaysFolderIsAvailable(PKNoFolder);
283             int hRestant = tbl_folders
284                     .getRemainingHoursFolderIsAvailable(PKNoFolder);
285             restant = (jRestant > 0) ? "<span style=\"color:green\">"
286                     + jRestant + " " + Global.i("CAPTION_DAYS").toLowerCase()
287                     + "</span>" : "<span style=\"color:orange\">"
288                     + (hRestant + 1) + " "
289                     + Global.i("CAPTION_HOURS").toLowerCase() + "</span>";
290
```



```
291             if (jRestant == 0 && hRestant <= 0) {
292                 restant = "<span style=\"color:red\">"
293                         + Global.i("CAPTION_ARCHIVAGE_AT")
294                         + " "
295                         + Utilities
296                                 .zeroFill(Integer.parseInt(Utilities
297                                         .formatSQLDate(
298                                                 folder.getCreation_date(), "HH")) +
    1)
299                         + ":00</span>";
300             }
301
302             contentLabel += "<table>"
303                     + "<tr><th>"
304                     + Global.i("CAPTION_CREATION")
305                     + "</th> <th>"
306                     + Global.i("CAPTION_EXPIRATION")
307                     + "</th> <th>"
308                     + Global.i("CAPTION_RESTANT")
309                     + "</th></tr>"
310                     + "<tr><td>"
311                     + Utilities.formatSQLDateWtText(folder.getCreation_date(),
312                             "dd MMMM YY")
313                     + "</td> <td>"
314                     + Utilities.formatSQLDateWtText(
315                             folder.getExpiration_date(), "dd MMMM YY")
316                     + "</td> <td>" + restant + "</td></tr>" + "</table>";
317         }
318
319         contentLabel += "</div>";
320
321         if (folder.getDescription() != null
322                 && folder.getDescription().length() > 0) {
323             contentLabel += "<div class=\"description\">"
324                     + Utilities.htmlentities(folder.getDescription())
325                     + "</div>";
326         }
327
328         // Récupération des destinataires qui ont été ajoutés au dossier
329         ArrayList<Contact> recipient = new ArrayList<Contact>();
330         ResultSet dest = Sql.query(tbl_recipients
331                 .getSelectRecipientsFromFolder(PKNoFolder));
332         try {
333             while (dest.next()) {
334                 recipient.add(new Contact(dest.getInt("PKNoContact"), dest
335                         .getString("contact_name"), dest
336                         .getString("contact_mail"), dest
337                         .getBoolean("contact_internal")));
338             }
339         } catch (SQLException e) {
340             e.printStackTrace();
341         }
342
343         if (recipient.size() > 0) {
344             contentLabel += "<p class=\"title\">"
345                     + Global.i("CAPTION_RECIPIENTS") + "</p>";
346         } else {
347             contentLabel += "<br><div align=\"center\"><b>"
348                     + Global.i("CAPTION_NO_RECIPIENT") + "</b></div>";
349         }
350
351         contentLabel += "</div>";
```

```
352
353         // Affichage du contenu imbriqué jusqu'ici (informations sur le dossier)
354         Label lblInfos = new Label(contentLabel, Label.CONTENT_XHTML);
355         grdInfos.addComponent(lblInfos, 0, 0);
356
357         // Affichage des boutons des destinataires
358         VerticalLayout vlRecipients = new VerticalLayout();
359         vlRecipients.setWidth("100%");
360         vlRecipients.setStyleName("vl-recipients");
361         grdInfos.addComponent(vlRecipients, 0, 1);
362         grdInfos.setRowExpandRatio(1, 100);
363
364         if (recipient.size() > 0) {
365             ArrayList<Button> btRecipient = new ArrayList<Button>();
366             int i = 0;
367             // On parcours tout les destinataires
368             for (final Contact contact : recipient) {
369                 btRecipient.add(new Button(contact.getName()));
370                 btRecipient.get(i).setStyleName(Runo.BUTTON_SMALL);
371
372                 final ResultSet data = Sql
373                         .query("SELECT * FROM trans_journal_folders "
374                                 + "LEFT JOIN trans_files ON FKNoFile=PKNoFile "
375                                 + "WHERE FKNoContact=" + contact.getPK()
376                                 + " AND trans_journal_folders.FKNoFolder="
377                                 + folder.getPKNoFolder() + " "
378                                 + "ORDER BY joufo_date DESC");
379
380                 try {
381                     if (data.first()) { // Le contact a des actions.
382                         int numberActions = 0;
383                         data.last();
384                         numberActions = data.getRow();
385                         data.beforeFirst();
386
387                         btRecipient.get(i).setIcon(
388                                 new ThemeResource(Global.PATH_THEME_RESSOURCES
389                                         + "play.png"));
390                         btRecipient.get(i).addStyleName("bt-active");
391                         btRecipient.get(i).setDescription(
392                                 numberActions + " "
393                                         + Global.i("CAPTION_LOG_ACTIONS"));
394                         btRecipient.get(i).addListener(new ClickListener() {
395                             public void buttonClick(ClickEvent event) {
396                                 Window winStat = new Window(Global
397                                         .i("CAPTION_LOG_ACTIONS_FOR")
398                                         + " \""
399                                         + contact.getName() + "\"");
400                                 CccvsTransfert.mainWindow.addWindow(winStat);
401                                 winStat.center();
402                                 winStat.setWidth("740px");
403                                 winStat.setHeight("510px");
404                                 winStat.setResizable(false);
405                                 winStat.setModal(true);
406
407                                 GridLayout grdStat = new GridLayout(2, 1);
408                                 grdStat.setStyleName("winStat");
409                                 grdStat.setSizeFull();
410                                 grdStat.setColumnExpandRatio(1, 100);
411                                 winStat.addComponent(grdStat);
412
413                                 VerticalLayout vlLeft = new VerticalLayout();
```



```
414                                 vlLeft.setSizeFull();
415                                 vlLeft.setWidth("220px");
416                                 grdStat.addComponent(vlLeft, 0, 0);
417
418                                 VerticalLayout vlRight = new VerticalLayout();
419                                 vlRight.setSizeFull();
420                                 grdStat.addComponent(vlRight, 1, 0);
421
422                                 Label lblResume = new Label(Global
423                                         .i("CAPTION_RESUME"));
424                                 lblResume.setStyleName("title");
425                                 vlLeft.addComponent(lblResume);
426
427                                 Label lblDetailled = new Label(Global
428                                         .i("CAPTION_DETAILLED"));
429                                 lblDetailled.setStyleName("title");
430                                 vlRight.addComponent(lblDetailled);
431
432                                 Table tblStatExpanded = new Table();
433                                 tblStatExpanded.setSizeFull();
434                                 tblStatExpanded
435                                         .setColumnHeaderMode(Table.COLUMN_HEADER_MO
    DE_HIDDEN);
436                                 tblStatExpanded.addContainerProperty(
437                                         "action_img", Embedded.class, null, "",
438                                         null, Table.ALIGN_CENTER);
439                                 tblStatExpanded.addContainerProperty(
440                                         "action_type", String.class, null,
441                                         Global.i("CAPTION_OBJECT"), null, null);
442                                 tblStatExpanded.addContainerProperty(
443                                         "action_date", String.class, null,
444                                         Global.i("CAPTION_DATE"), null, null);
445                                 tblStatExpanded
446                                         .setColumnWidth("action_img", 23);
447                                 tblStatExpanded.setColumnWidth("action_date",
448                                         140);
449                                 tblStatExpanded.setColumnWidth("action_type",
450                                         260);
451                                 vlRight.addComponent(tblStatExpanded);
452
453                                 int consultations = 0;
454                                 int archive = 0;
455                                 int download = 0;
456                                 try {
457                                     data.beforeFirst();
458                                     while (data.next()) {
459                                         if (data.getString("joufo_action")
460                                                 .equals("download")) {
461                                             String filename = (data
462                                                     .getString("FKNoFile") != null)
    ? data
463                                                     .getString("file_rename")
464                                                     :
    Global.i("CAPTION_ARCHIVE_OF_FOLDER");
465                                             if (data.getString("FKNoFile") != null)
    {
466                                                 archive++;
467                                             } else {
468                                                 download++;
469                                             }
470                                             boolean isFile = (data
471                                                     .getString("FKNoFile") != null)
```

```
   ? true
                                                    : false;

                                            Embedded imgIllustr = new Embedded(
                                                    null,
                                                    new ThemeResource(
                                                            (isFile) ?
   Global.IMG_DOWNLOAD
                                                                    :
   Global.IMG_ARCHIVE_DOWNLOAD));
                                            tblStatExpanded.addItem(
                                                    new Object[] {
                                                            imgIllustr,
                                                            filename,
                                                            Utilities
                                                                    .formatSQLDate(
                                                                            data.ge
   tString("joufo_date"),
                                                                            "dd
   MMMM YY HH:mm:ss") },
                                                            tblStatExpanded.size());
                                                } else {
                                                    consultations++;
                                                    Embedded imgIllustr = new Embedded(
                                                            null, new ThemeResource(
                                                                    Global.IMG_SEARCH));
                                                    tblStatExpanded.addItem(
                                                            new Object[] {
                                                                    imgIllustr,
                                                                    Global.i("CAPTION_CONSU
   LTATION"),
                                                                    Utilities
                                                                            .formatSQLDate(
                                                                                    data.ge
   tString("joufo_date"),
                                                                                    "dd
   MMMM YY HH:mm:ss") },
                                                            tblStatExpanded.size());
                                                }
                                            }
                                        } catch (UnsupportedOperationException e) {
                                            e.printStackTrace();
                                        } catch (SQLException e) {
                                            e.printStackTrace();
                                        }

                                        Table tblStatResume = new Table();
                                        tblStatExpanded.setSizeFull();
                                        tblStatResume
                                                .setColumnHeaderMode(Table.COLUMN_HEADER_MO
   DE_HIDDEN);
                                        tblStatResume.addContainerProperty(
                                                "action_img", Embedded.class, null, "",
                                                null, Table.ALIGN_CENTER);
                                        tblStatResume.addContainerProperty(
                                                "action_type", String.class, null,
                                                Global.i("CAPTION_ACTION"), null, null);
                                        tblStatResume.addContainerProperty(
                                                "action_num", Integer.class, null, "",
                                                null, null);
                                        tblStatResume.setColumnWidth("action_img", 25);
                                        tblStatResume.setColumnWidth("action_num", 25);
```



```
                                        tblStatResume.addItem(
                                                new Object[] {
                                                        new Embedded(
                                                                null,
                                                                new ThemeResource(
                                                                        Global.IMG_SEARCH)),
                                                        Global.i("CAPTION_CONSULTATION"),
                                                        consultations }, 1);
                                        tblStatResume.addItem(new Object[] {
                                                new Embedded(null, new ThemeResource(
                                                        Global.IMG_ARCHIVE_DOWNLOAD)),
                                                Global.i("CAPTION_ARCHIVE_OF_FOLDER"),
                                                archive }, 2);
                                        tblStatResume.addItem(
                                                new Object[] {
                                                        new Embedded(
                                                                null,
                                                                new ThemeResource(
                                                                        Global.IMG_DOWNLOAD)
   ),
                                                        Global.i("CAPTION_DOWNLOADS"),
                                                        download }, 3);
                                        vlLeft.addComponent(tblStatResume);
                                    }
                                });

                            } else { // Si le contact n'a aucune action
                                btRecipient.get(i).addStyleName("bt-non-active");
                                btRecipient.get(i).addListener(new ClickListener() {
                                    public void buttonClick(ClickEvent event) {
                                        CccvsTransfert.mainWindow.showNotification(
                                                contact.getName(),
                                                Global.i("CAPTION_FOLDER_NOT_WATCHED"),
                                                Notification.TYPE_WARNING_MESSAGE);
                                    }
                                });
                            }
                        } catch (SQLException e) {
                            e.printStackTrace();
                        }

                        vlRecipients.addComponent(btRecipient.get(i));
                        i++;
                    }
                }

                Label lblStatus = new Label(Global.i("CAPTION_FREE"));
                lblStatus.setStyleName("folder-status");
                if (folder.getArchive()) {
                    lblStatus.addStyleName("archive");
                    lblStatus.setValue(Global.i("CAPTION_ARCHIVED"));
                }
                grdInfos.addComponent(lblStatus, 0, 2);
                grdInfos.setComponentAlignment(lblStatus, Alignment.BOTTOM_CENTER);

                return body;
            }

            /**
             * Retourne la liste des fichiers contenus dans le dossier sélectionné
             *
             * @return Liste des fichiers du dossier
```

```
     */
    private static ArrayList<Files> getFileList() {
        ArrayList<Files> files = new ArrayList<Files>();
        ResultSet data = Sql.query(tbl_files
                .getSelectAllFilesFromFolder(FKNoFolder));
        try {
            while (data.next()) {
                files.add(new Files(data.getInt("PKNoFile"), data
                        .getInt("FKNoFolder"), data.getString("file_name"),
                        data.getString("file_rename"), data
                                .getLong("file_size"), data
                                .getString("file_extension"), data
                                .getString("file_description"), Utilities
                                .formatSize(data.getLong("file_size"))
                // Utilities.getImgFromExtension(data.getString("file_extension"))
                ));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return files;
    }
}
```



```
package modules;

import global.Global;

/**
 * Module qui permet la récupération d'un lien de fichier ou d'un lien d'une
 * archive de dossier.<br>
 * L'utilisateur peut ainsi télécharger lui même ses propres fichiers ou
 * partager un fichier avec n'importe qui d'autre en lui donnant uniquement
 * l'URL vers le fichier.
 *
 * @author Dominique Roduit
 *
 */
public class GetLinkModule extends CustomComponent {
    /** Stockage des objets conservés en global **/
    private GlobalObjects global = CccvsTransfert.getGlobalMethod();
    /** Layout racine du module **/
    private VerticalLayout mainLayout = new VerticalLayout();

    /**
     * Affichage des composants (TextArea, boutons)
     *
     * @param filename
     *            Nom du fichier sur le disque
     */
    public GetLinkModule(String filename) {
        mainLayout.setSizeFull();
        mainLayout.setSpacing(true);

        TextArea txtLink = new TextArea();
        txtLink.setWidth("100%");
        txtLink.setHeight("50px");
        txtLink.setValue(Utilities.getURLForFile(filename));
        txtLink.selectAll();
        txtLink.focus();
        txtLink.setReadOnly(true);
        txtLink.setStyleName("get-link-textarea");
        mainLayout.addComponent(txtLink);

        HorizontalLayout buttons = new HorizontalLayout();
        buttons.setSpacing(true);
        mainLayout.addComponent(buttons);

        Button btAnnuler = new Button(Global.i("CAPTION_CLOSE"),
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        closeWindow();
                    }
                });

        buttons.addComponent(btAnnuler);
        buttons.setComponentAlignment(btAnnuler, Alignment.MIDDLE_RIGHT);

        mainLayout.setComponentAlignment(buttons, Alignment.MIDDLE_RIGHT);

        setCompositionRoot(mainLayout);
    }

    /**
     * Fermeture de la fenêtre flottante
     */
```

```
    private void closeWindow() {
        (global.getWinGetLink().getParent()).removeWindow(global
                .getWinGetLink());
    }
}
```



```
package modules;

import global.Global;

/**
 * Redistribution des e-mail pour l'accès à un dossier.<br>
 * Ce module peut être utile si un des contacts dit ne pas avoir reçu le mail
 * pour X raisons, il permet de réenvoyer le lien crypté.<br>
 * Par contre la date d'expiration est conservée et ne change pas.
 *
 * @author Dominique Roduit
 *
 */
public class ReMailLinkModule extends CustomComponent {
    /** Stockage des objets déclarés en global **/
    private GlobalObjects global = CccvsTransfert.getGlobalMethod();
    /** Clé primaire du dossier **/
    private int PKNoFolder = 0;
    /** Layout racine du module **/
    private VerticalLayout mainLayout = new VerticalLayout();
    /** Contient la liste des contacts inclus dans le dossier **/
    private BeanItemContainer<Contact> cntContacts = new
 BeanItemContainer<Contact>(
            Contact.class);
    /** Composant table qui contient les contacts **/
    private Table tblContacts;
    /** Contient les informations sur le dossier **/
    private Folder folder;

    /**
     * Affichage des composants du module (Tableau, boutons, textes, ...)
     *
     * @param pKNoFolder
     *            Clé primaire du dossier concerné
     */
    public ReMailLinkModule(int pKNoFolder) {
        this.PKNoFolder = pKNoFolder;

        ResultSet fold = Sql
                .query(tbl_folders.getSelectFolderInfos(PKNoFolder));
        try {
            if (fold.first()) {
                folder = new Folder(PKNoFolder, fold.getInt("FKNoUser"),
                        fold.getString("folder_name"),
                        fold.getString("folder_description"),
                        fold.getString("folder_creation_date"),
                        fold.getString("folder_expiration"),
                        fold.getBoolean("folder_archive"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        mainLayout.setSizeFull();
        mainLayout.setSpacing(true);

        Label lblInstructions = new Label(
                Global.i("CAPTION_SEND_BACK_INSTRUCTIONS"));
        mainLayout.addComponent(lblInstructions);

        ArrayList<Contact> contactList = getContactList();
        cntContacts.addAll(contactList);
```

```
95
96          tblContacts = new Table();
97          tblContacts.setSizeFull();
98          tblContacts.setHeight("300px");
99          tblContacts.setSelectable(true);
100         tblContacts.setMultiSelect(true);
101         tblContacts.setMultiSelectMode(MultiSelectMode.SIMPLE);
102         tblContacts.setContainerDataSource(cntContacts);
103         tblContacts.setVisibleColumns(new String[] { "name", "mail" });
104         tblContacts.setColumnHeaders(new String[] { Global.i("CAPTION_NAME"),
105                 Global.i("CAPTION_ADRESSE_EMAIL") });
106         tblContacts.addGeneratedColumn("contact_internal",
107                 new Table.ColumnGenerator() {
108                     public Object generateCell(Table source, Object itemId,
109                             Object columnId) {
110                         Contact contact = (Contact) itemId;
111                         String imgSrc = (contact.isInternal()) ? "home.png"
112                                 : "icon-contact.png";
113                         Embedded imgContact = new Embedded(null,
114                                 new ThemeResource(Global.PATH_THEME_RESSOURCES
115                                         + imgSrc));
116                         return imgContact;
117                     }
118                 });
119         tblContacts.setColumnHeader("contact_internal", "");
120         tblContacts.setColumnWidth("contact_internal", 24);
121         tblContacts.setColumnAlignment("contact_internal", Table.ALIGN_CENTER);
122         tblContacts.setVisibleColumns(new String[] { "contact_internal",
123                 "name", "mail" });
124         mainLayout.addComponent(tblContacts);
125
126         HorizontalLayout buttons = new HorizontalLayout();
127         buttons.setSpacing(true);
128         mainLayout.addComponent(buttons);
129
130         Button btAnnuler = new Button(Global.i("CAPTION_CANCEL"),
131                 new Button.ClickListener() {
132                     public void buttonClick(ClickEvent event) {
133                         closeWindow();
134                     }
135                 });
136
137         Button btSend = new Button(Global.i("CAPTION_SEND"),
138                 new Button.ClickListener() {
139                     public void buttonClick(ClickEvent event) {
140                         String contacts = "<ul style=\"font-size:9pt\">";
141                         Set<?> value = (Set<?>) tblContacts.getValue();
142                         if (null == value || value.size() == 0) {
143                             getWindow().showNotification(null,
144                                     Global.i("CAPTION_MUST_SELECT_CONTACT"),
145                                     Notification.TYPE_WARNING_MESSAGE);
146                         } else {
147                             // On parcours les contacts sélectionnés
148                             for (Object val : value) {
149                                 Contact c = (Contact) val;
150                                 System.out.println(c.getPK() + " - "
151                                         + c.getName() + " - " + c.getMail());
152                                 System.out.println(PKNoFolder + " - "
153                                         + folder.getExpiration_date() + " - "
154                                         + folder.getDescription());
155
156                                 // Envoi du mail aux contacts sélectionnés
```



```
157                                 int expiration = tbl_folders
158                                         .getRemainingDaysFolderIsAvailable(PKNoFold
    er);
159                                 Mailer.sendDownloadLink(c.getMail(),
160                                         PKNoFolder, c.getPK(),
161                                         folder.getExpiration_date(),
162                                         expiration, folder.getDescription(),
163                                         folder.getName());
164
165                                 contacts += "<li>" + c.getName() + "</li>";
166                             }
167                             contacts += "</ul>";
168
169                             // Affichage de la notification
170                             CccvsTransfert.mainWindow.showNotification(Global
171                                     .i("CAPTION_LINK_REDISTRIBUTED"),
172                                     Global.i("CAPTION_MAIL_SEND_BACK_TO")
173                                             .replace("%contacts%", contacts),
174                                     Notification.TYPE_TRAY_NOTIFICATION, true);
175
176                             // Fermeture de la fenêtre
177                             closeWindow();
178                         }
179                     }
180                 });
181
182         buttons.addComponent(btAnnuler);
183         buttons.setComponentAlignment(btAnnuler, Alignment.MIDDLE_RIGHT);
184
185         buttons.addComponent(btSend);
186         buttons.setComponentAlignment(btSend, Alignment.MIDDLE_RIGHT);
187
188         mainLayout.setComponentAlignment(buttons, Alignment.MIDDLE_RIGHT);
189
190         setCompositionRoot(mainLayout);
191     }
192
193     /**
194      * Fermeture de la fenêtre flottante
195      */
196     private void closeWindow() {
197         (global.getWinReMailLink().getParent()).removeWindow(global
198                 .getWinReMailLink());
199     }
200
201     /**
202      * Récupération de la liste des contacts invités à consulter le dossier
203      *
204      * @return Liste des contacts inclus dans le dossier
205      */
206     private ArrayList<Contact> getContactList() {
207         ResultSet contact = Sql.query(tbl_contacts
208                 .getSelectContactsFromFolder(PKNoFolder));
209         ArrayList<Contact> contactList = new ArrayList<Contact>();
210
211         try {
212             while (contact.next()) {
213                 contactList.add(new Contact(contact.getInt("PKNoContact"),
214                         contact.getInt("FKNoUser"), contact
215                                 .getString("contact_name"), contact
216                                 .getString("contact_mail"), contact
217                                 .getString("contact_creation_date"), contact
```

```
                                .getBoolean("contact_internal")));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return contactList;
    }

}
```



```
package modules;

import main.CccvsTransfert;

/**
 * Page de gestion des transferts.<br>
 * Elle affiche un bouton dans le ruban du menu pour permettre la création d'un
 * nouveau dossier.<br>
 * La fenêtre d'assistance à la navigation est implémentée dans cette class.<br>
 * Lorsqu'un dossier est affiché, c'est la class {@link FolderModule} qui est
 * implémentée.
 *
 * @author Dominique Roduit
 *
 */
public class TransfertModule {
    /**
     * Contient toutes les instances qui doivent être accessibles dans toute
     * l'application
     **/
    private static GlobalObjects global = CccvsTransfert.getGlobalMethod();
    /** Instance de la fenêtre principale **/
    private static Window mainWindow = CccvsTransfert.mainWindow;
    /** Layout Global de l'application **/
    private static MainLayout mainLayout = global.getMainLayout();
    /** Bouton d'ajout d'un contact **/
    private static Button btAddFolder;
    /** Table contenant la liste des contacts **/
    private static ContactTable tblFolder;
    /** Contient une instance de la fenêtre pour la création d'un dossier **/
    private static Window winFolder;

    /**
     * Chargement des boutons dans le ruban du wrapper principal.<br>
     * <b>Bouton</b> "Nouveau dossier"
     */
    public static Button getBodyRibbonContent() {
        System.out.println(Utilities.getCurrentDate()
                + " -- [Chargement du module] Ruban");

        btAddFolder = new Button(Global.i("CAPTION_NEW_FOLDER"));
        btAddFolder.setIcon(new ThemeResource(Global.IMG_FOLDER_ADD));
        btAddFolder.focus();

        global.setBtAddFolder(btAddFolder);

        btAddFolder.addListener(new Button.ClickListener() {
            @Override
            public void buttonClick(ClickEvent event) {
                if (tbl_contacts.getNumberOfContact() < 1) {
                    CccvsTransfert.mainWindow.showNotification(
                            Global.i("CAPTION_ERROR"),
                            Global.i("CAPTION_ERROR_NO_CONTACT"),
                            Notification.TYPE_WARNING_MESSAGE);
                    CccvsTransfert.loadModule(new Module(ContactModule
                            .getBodyRibbonContent(), ContactModule
                            .getBodyContent()));
                    AccordionMenu.getMenu().setSelectedTab(1);
                } else {
                    winFolder = new Window(Global.i("CAPTION_TRANSFERT")
                            + " - " + Global.i("TITLE_WINDOW_CREATE_FOLDER"));
                    winFolder.setIcon(new ThemeResource(
```

```
                            Global.PATH_THEME_RESSOURCES
                                    + "folder-horizontal.png"));
                    winFolder.setModal(true);
                    winFolder.setWidth("440px");
                    winFolder.setHeight("255px");
                    winFolder.setResizable(false);
                    winFolder.setStyleName(Runo.WINDOW_DIALOG);
                    CccvsTransfert.mainWindow.addWindow(winFolder);
                    global.setWindowFolder(winFolder);

                    // Chargement du premier module du wizard
                    WizFoldNew wzrTrans;
                    wzrTrans = new WizFoldNew();
                    winFolder.addComponent(wzrTrans);
                }
            }
        });

        return btAddFolder;
    }

    /**
     * Insertion du contenu de la page dans le corps de l'application.<br>
     * <br>
     * <b>Contenu :</b><br>
     * <ul>
     * <li>Assistance à la première utilisation</li>
     * <li>Page de démarrage</li>
     * </ul>
     */
    public static VerticalLayout getBodyContent() {
        System.out.println(Utilities.getCurrentDate()
                + " -- [Chargement du module] Body");

        VerticalLayout body = new VerticalLayout();
        body.setSizeFull();

        Embedded imgIllustr = new Embedded(Global.getParm("FACTORY_NAME"),
                new ThemeResource(Global.PATH_THEME_RESSOURCES
                        + "home_illustration.png"));
        imgIllustr.addStyleName("schema-illustration");
        body.addComponent(imgIllustr);
        body.setComponentAlignment(imgIllustr, Alignment.TOP_CENTER);

        String content = "";
        String imgTrue = " <img src=\"" + Global.PATH_THEME_RESSOURCES_HTML
                + "true.png\" />";
        String imgFalse = " <img src=\"" + Global.PATH_THEME_RESSOURCES_HTML
                + "false.png\" />";

        // S'il n'existe pas encore de dossier
        int nbFolder = tbl_folders.getNumberOfFolders();
        int nbContact = tbl_contacts.getNumberOfContact();

        if (nbFolder < 1 || nbContact < 1) {
            content += Global.i("CONTENT_HOME_TITLE");
            content += Global.i("CONTENT_HOME_OBJ1");
            content += (nbContact < 1) ? imgFalse : imgTrue;
            content += Global.i("CONTENT_HOME_OBJ2");
            content += (nbFolder < 1) ? imgFalse : imgTrue;
        } else {
            content += Global.i("CONTENT_HOME_TITLE");
```



```
            content += Global.i("CONTENT_HOME_OBJ3");
            content += Global.i("CONTENT_HOME_OBJ2");
        }

        Label lblHome = new Label(content, Label.CONTENT_XHTML);
        lblHome.addStyleName("align-center");
        body.addComponent(lblHome);
        body.setComponentAlignment(lblHome, Alignment.TOP_CENTER);

        return body;
    }
}
```

```java
package modules.admin;

import java.sql.ResultSet;

/**
 * Module disponible uniquement par les administrateurs. Il permet la
 * visualisation rapide de toutes les actions qui ont été journalisées.
 *
 * @author Dominique Roduit
 *
 */
public class AdminJournModule {
    /**
     * Contient toutes les instances qui doivent être accessibles dans toute
     * l'application
     **/
    private static GlobalObjects global = CccvsTransfert.getGlobalMethod();
    /** Instance de la fenêtre principale **/
    private static Window mainWindow = CccvsTransfert.mainWindow;
    /** Layout Global de l'application **/
    private static MainLayout mainLayout = global.getMainLayout();
    /** Table contenant la liste des contacts **/
    private static Table tblActions;
    /**
     * Conteneur de la liste des actions journalisées pour les connexions et
     * validations
     **/
    private static BeanItemContainer<JournalCon> cntActions = new
  BeanItemContainer<JournalCon>(
            JournalCon.class);
    /** Boutons du menu dans le ruban **/
    private static ArrayList<Button> btMenu;
    /** Layout horizontal qui contient les boutons de filtre **/
    private static HorizontalLayout hlFilter;

    /**
     * Chargement du bouton de retour et du titre dans le ruban du corps de la
     * page
     */
    public static VerticalLayout getBodyRibbonContent() {
        System.out.println(Utilities.getCurrentDate()
                + " -- [Chargement du module] Ruban");

        VerticalLayout vlMain = new VerticalLayout();
        vlMain.setSizeFull();
        vlMain.setSpacing(true);

        HorizontalLayout hlRibbon = new HorizontalLayout();
        hlRibbon.setSizeFull();
        hlRibbon.setSpacing(true);
        vlMain.addComponent(hlRibbon);

        hlFilter = new HorizontalLayout();
        hlFilter.setSizeFull();
        hlFilter.setSpacing(true);
        vlMain.addComponent(hlFilter);

        btMenu = new ArrayList<Button>();

        // Bouton Connexions
        final ArrayList<Button> filterConnexions = new ArrayList<Button>();
        filterConnexions.add(new Button(Global.i("CAPTION_PASSWORD_FAILED"),
```



```java
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        loadTableWithConnexions("con_wrong_pass");
                    }
                }));
        filterConnexions.add(new Button(Global
                .i("CAPTION_CONNECTION_SUCCESSFUL"),
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        loadTableWithConnexions("con_success");
                    }
                }));
        filterConnexions.add(new Button(Global.i("CAPTION_INVALID_INFOS"),
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        loadTableWithConnexions("con_invalid_infos");
                    }
                }));
        filterConnexions.add(new Button(Global
                .i("CAPTION_EMAIL_INVALID_FORMAT"), new Button.ClickListener() {
            public void buttonClick(ClickEvent event) {
                loadTableWithConnexions("con_invalid_mail");
            }
        }));
        filterConnexions.add(new Button(Global.i("CAPTION_NEW_USER"),
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        loadTableWithConnexions("con_new_user");
                    }
                }));
        filterConnexions.add(new Button(Global.i("CAPTION_INVALIDATE_ACCOUNT"),
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        loadTableWithConnexions("con_no_validate");
                    }
                }));

        btMenu.add(new Button(Global.i("CAPTION_CONNECTIONS"),
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        addFilters(filterConnexions);
                        loadTableWithConnexions("");
                    }
                }));
        btMenu.get(0).setStyleName(Runo.BUTTON_DEFAULT);
        btMenu.get(0).setIcon(
                new ThemeResource(Global.PATH_THEME_RESSOURCES + "unlock.png"));

        // Bouton Validation des comptes
        final ArrayList<Button> filderValidation = new ArrayList<Button>();
        filderValidation.add(new Button(Global.i("CAPTION_ERRORS"),
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        loadTableWithValidations("validation_error");
                    }
                }));
        filderValidation.add(new Button(Global.i("CAPTION_RECCURENCES"),
                new Button.ClickListener() {
                    public void buttonClick(ClickEvent event) {
                        loadTableWithValidations("validation_recurrent");
                    }
                }));
```

```
173         filderValidation.add(new Button(Global.i("CAPTION_VALIDES"),
174                 new Button.ClickListener() {
175                     public void buttonClick(ClickEvent event) {
176                         loadTableWithValidations("validation_success");
177                     }
178                 }));
179 
180         btMenu.add(new Button(Global.i("CAPTION_ACCOUNTS_VALIDATION"),
181                 new Button.ClickListener() {
182                     public void buttonClick(ClickEvent event) {
183                         addFilters(filderValidation);
184                         loadTableWithValidations("");
185                     }
186                 }));
187         btMenu.get(1).setIcon(
188                 new ThemeResource(Global.PATH_THEME_RESSOURCES
189                         + "tick-shield.png"));
190 
191         // Bouton Téléchargements
192         final ArrayList<Button> filterDownload = new ArrayList<Button>();
193         filterDownload.add(new Button(Global.i("CAPTION_FOLDERS_ARCHIVE"),
194                 new Button.ClickListener() {
195                     public void buttonClick(ClickEvent event) {
196                         loadTableWithDownloads("trans_journal_folders.FKNoFile IS
    NULL AND joufo_action='download'");
197                     }
198                 }));
199         filterDownload.add(new Button(Global.i("CAPTION_CONSULTATIONS"),
200                 new Button.ClickListener() {
201                     public void buttonClick(ClickEvent event) {
202                         loadTableWithDownloads("joufo_action='consult' AND
    trans_journal_folders.FKNoFile IS NULL");
203                     }
204                 }));
205         filterDownload.add(new Button(Global.i("CAPTION_FILES_DOWNLOADS"),
206                 new Button.ClickListener() {
207                     @Override
208                     public void buttonClick(ClickEvent event) {
209                         loadTableWithDownloads("joufo_action='download' AND
    trans_journal_folders.FKNoFile IS NOT NULL");
210                     }
211                 }));
212 
213         btMenu.add(new Button(Global.i("CAPTION_DOWNLOADS"),
214                 new Button.ClickListener() {
215                     public void buttonClick(ClickEvent event) {
216                         addFilters(filterDownload);
217                         loadTableWithDownloads("");
218                     }
219                 }));
220         btMenu.get(2).setIcon(
221                 new ThemeResource(Global.PATH_THEME_RESSOURCES + "down.png"));
222 
223         // Insertion des boutons
224         for (Button button : btMenu) {
225             button.setWidth("100%");
226             button.addListener(new Button.ClickListener() {
227                 public void buttonClick(ClickEvent event) {
228                     deselectAllButtons();
229                     event.getButton().setStyleName(Runo.BUTTON_DEFAULT);
230                 }
231             });
```



```
232             hlRibbon.addComponent(button);
233             hlRibbon.setComponentAlignment(button, Alignment.MIDDLE_CENTER);
234         }
235 
236         // Ajout des composants de filtrage de l'information
237         addFilters(filterConnexions);
238 
239         return vlMain;
240     }
241 
242     /**
243      * Ajout des boutons de filtrage
244      *
245      * @param btList
246      *            Liste des boutons de filtrage
247      */
248     private static void addFilters(ArrayList<Button> btList) {
249         hlFilter.removeAllComponents();
250         for (Button bt : btList) {
251             bt.setStyleName(Runo.BUTTON_SMALL);
252             hlFilter.addComponent(bt);
253         }
254     }
255 
256     /**
257      * Insertion du contenu dans le corps de la page.<br>
258      * Le contenu comprend le texte, extrait de la base de données ainsi qu'une
259      * image illustrative.
260      */
261     public static VerticalLayout getBodyContent() {
262         System.out.println(Utilities.getCurrentDate()
263                 + " -- [Chargement du module] Body");
264 
265         VerticalLayout vlBody = new VerticalLayout();
266         vlBody.setSizeFull();
267 
268         loadTableWithConnexions("");
269 
270         tblActions = new Table();
271         vlBody.addComponent(tblActions);
272         tblActions.setSizeFull();
273         tblActions.setSelectable(true);
274         tblActions.setContainerDataSource(cntActions);
275         tblActions.setVisibleColumns(new String[] { "joco_action", "joco_mail",
276                 "joco_browser", "joco_date", "joco_os", "joco_ip", });
277 
278         // Génération de la colonne "Admin"
279         tblActions.addGeneratedColumn("action", new Table.ColumnGenerator() {
280             public Object generateCell(Table source, Object itemId,
281                     Object columnId) {
282                 JournalCon action = (JournalCon) itemId;
283                 String imgAction = "";
284                 if (action.getJoco_action().equals("con_invalid_infos")) {
285                     imgAction = "ui-text-field-password-red.png";
286                 } else if (action.getJoco_action().equals("con_invalid_mail")) {
287                     imgAction = "mail--exclamation.png";
288                 } else if (action.getJoco_action().equals("con_new_user")) {
289                     imgAction = "plus.png";
290                 } else if (action.getJoco_action().equals("con_no_validate")) {
291                     imgAction = "mail-forward-all.png";
292                 } else if (action.getJoco_action().equals("con_wrong_pass")
293                         || action.getJoco_action().equals("validation_error")) {
```

```
294                     imgAction = "exclamation-red.png";
295                 } else if (action.getJoco_action().equals("con_success")
296                         || action.getJoco_action().equals("validation_success")) {
297                     imgAction = "tick.png";
298                 } else if (action.getJoco_action().equals(
299                         "validation_recurrent")) {
300                     imgAction = "reload.png";
301                 } else if (action.getJoco_action().equals("consult")) {
302                     imgAction = "search.png";
303                 } else if (action.getJoco_action().equals("download")) {
304                     imgAction = "down.png";
305                 } else if (action.getJoco_action().equals("archive")) {
306                     imgAction = "archive_down.png";
307                 }
308                 return new Embedded(null, new ThemeResource(
309                         Global.PATH_THEME_RESSOURCES + imgAction));
310             }
311         });
312         tblActions.setColumnWidth("action", 25);
313         tblActions.setColumnAlignment("action", Table.ALIGN_CENTER);
314 
315         // Génération de la colonne "Navigateur"
316         tblActions.addGeneratedColumn("browser", new Table.ColumnGenerator() {
317             public Object generateCell(Table source, Object itemId,
318                     Object columnId) {
319                 JournalCon action = (JournalCon) itemId;
320                 String imgBrowser = "browser.png";
321                 if (action.getJoco_browser().indexOf("Internet Explorer") > -1) {
322                     imgBrowser = "ie.png";
323                 } else if (action.getJoco_browser().indexOf("Firefox") > -1) {
324                     imgBrowser = "firefox.png";
325                 } else if (action.getJoco_browser().indexOf("Chrome") > -1) {
326                     imgBrowser = "chrome.png";
327                 } else if (action.getJoco_browser().indexOf("Opera") > -1) {
328                     imgBrowser = "opera.png";
329                 } else if (action.getJoco_browser().indexOf("Safari") > -1) {
330                     imgBrowser = "safari.png";
331                 }
332                 return new Embedded(null, new ThemeResource(
333                         Global.PATH_THEME_RESSOURCES + imgBrowser));
334             }
335         });
336         tblActions.setColumnWidth("browser", 25);
337         tblActions.setColumnAlignment("browser", Table.ALIGN_CENTER);
338 
339         // Génération de la colonne "OS"
340         tblActions.addGeneratedColumn("os", new Table.ColumnGenerator() {
341             public Object generateCell(Table source, Object itemId,
342                     Object columnId) {
343                 JournalCon action = (JournalCon) itemId;
344                 String imgOS = "other.png";
345                 if (action.getJoco_os().indexOf("Windows") > -1) {
346                     imgOS = "windows.png";
347                 } else if (action.getJoco_os().indexOf("Mac OSX") > -1) {
348                     imgOS = "mac.png";
349                 } else if (action.getJoco_os().indexOf("Linux") > -1) {
350                     imgOS = "linux.png";
351                 }
352                 return new Embedded(null, new ThemeResource(
353                         Global.PATH_THEME_RESSOURCES + imgOS));
354             }
355         });
```



```
356         tblActions.setColumnWidth("os", 25);
357         tblActions.setColumnAlignment("os", Table.ALIGN_CENTER);
358 
359         // Génération de la colonne "Action"
360         tblActions.addGeneratedColumn("action_text",
361                 new Table.ColumnGenerator() {
362                     public Object generateCell(Table source, Object itemId,
363                             Object columnId) {
364                         JournalCon action = (JournalCon) itemId;
365                         String txtAction = "-";
366                         if (action.getJoco_action().equals("con_invalid_infos")) {
367                             txtAction = Global.i("CAPTION_INVALID_INFOS");
368                         } else if (action.getJoco_action().equals(
369                                 "con_invalid_mail")) {
370                             txtAction = Global.i("CAPTION_EMAIL_INVALID");
371                         } else if (action.getJoco_action().equals(
372                                 "con_new_user")) {
373                             txtAction = Global.i("CAPTION_NEW_USER");
374                         } else if (action.getJoco_action().equals(
375                                 "con_no_validate")) {
376                             txtAction = Global.i("CAPTION_INVALIDATE_ACCOUNT");
377                         } else if (action.getJoco_action().equals(
378                                 "con_wrong_pass")) {
379                             txtAction = Global.i("CAPTION_PASSWORD_FAILED");
380                         } else if (action.getJoco_action()
381                                 .equals("con_success")) {
382                             txtAction = Global
383                                     .i("CAPTION_CONNECTION_SUCCESSFUL");
384                         } else if (action.getJoco_action().equals(
385                                 "validation_success")) {
386                             txtAction = Global.i("CAPTION_VALIDATION_SUCCESS");
387                         } else if (action.getJoco_action().equals(
388                                 "validation_error")) {
389                             txtAction = Global.i("CAPTION_VALIDATION_ERROR");
390                         } else if (action.getJoco_action().equals(
391                                 "validation_recurrent")) {
392                             txtAction = Global.i("CAPTION_REPEATED_VALIDATION");
393                         } else if (action.getJoco_action().equals("consult")) {
394                             txtAction = Global.i("CAPTION_CONSULTATION");
395                         } else if (action.getJoco_action().equals("download")
396                                 || action.getJoco_action().equals("archive")) {
397                             txtAction = Global.i("CAPTION_DOWNLOADING");
398                         }
399                         return txtAction;
400                     }
401                 });
402         tblActions.setColumnWidth("action_text", 130);
403         tblActions.setColumnAlignment("action_text", Table.ALIGN_CENTER);
404 
405         tblActions.setVisibleColumns(new String[] { "action", "browser", "os",
406                 "action_text", "joco_mail", "joco_date", "joco_ip",
407                 "joco_browser" });
408         tblActions.setColumnHeaders(new String[] { "", "", "",
409                 Global.i("CAPTION_ACTION"), Global.i("CAPTION_ADRESSE_EMAIL"),
410                 Global.i("CAPTION_DATE"), "IP", Global.i("CAPTION_BROWSER") });
411 
412         return vlBody;
413     }
414 
415     /**
416      * Remplissage de la table avec les connexions journalisées
417      *
```

```
418      * @param where
419      *            Valeur de la clause WHERE de la requête SQL
420      */
421     private static void loadTableWithConnexions(String where) {
422         cntActions.removeAllItems();
423         cntActions.addAll(getConnexionsList(0, where));
424 
425         if (tblActions != null) {
426             tblActions.setVisibleColumns(new String[] { "action", "browser",
427                     "os", "action_text", "joco_mail", "joco_date", "joco_ip",
428                     "joco_browser" });
429             tblActions
430                     .setColumnHeaders(new String[] { "", "", "",
431                             Global.i("CAPTION_ACTION"),
432                             Global.i("CAPTION_ADRESSE_EMAIL"),
433                             Global.i("CAPTION_DATE"), "IP",
434                             Global.i("CAPTION_BROWSER") });
435         }
436     }
437 
438     /**
439      * Remplissage de la table avec les validations de comptes journalisées
440      *
441      * @param where
442      *            Valeur de la clause WHERE de la requête SQL
443      */
444     private static void loadTableWithValidations(String where) {
445         cntActions.removeAllItems();
446         cntActions.addAll(getConnexionsList(1, where));
447         tblActions.setVisibleColumns(new String[] { "action", "browser", "os",
448                 "action_text", "joco_mail", "joco_date", "joco_ip",
449                 "joco_browser" });
450         tblActions.setColumnHeaders(new String[] { "", "", "",
451                 Global.i("CAPTION_ACTION"), Global.i("CAPTION_ADRESSE_EMAIL"),
452                 Global.i("CAPTION_DATE"), "IP", Global.i("CAPTION_BROWSER") });
453     }
454 
455     /**
456      * Remplissage de la table avec les journalisations de téléchargement
457      *
458      * @param where
459      *            Valeur de la clause WHERE de la requête SQL
460      */
461     private static void loadTableWithDownloads(String where) {
462         cntActions.removeAllItems();
463         cntActions.addAll(getDownloadsList(where));
464         tblActions
465                 .setVisibleColumns(new String[] { "action", "action_text",
466                         "joco_os", "joco_browser", "joco_ip", "joco_mail",
467                         "joco_date" });
468         tblActions.setColumnHeaders(new String[] { "",
469                 Global.i("CAPTION_ACTION"), Global.i("CAPTION_AUTHOR_FOLDER"),
470                 Global.i("CAPTION_FOLDER"), Global.i("CAPTION_FILENAME"),
471                 Global.i("CAPTION_CONTACT_ACTION_AFFECTED"),
472                 Global.i("CAPTION_DATE") });
473     }
474 
475     /**
476      * Récupération de la liste des utilisateurs
477      *
478      * @param type
479      *            0 = connexion, 1 = validations
```



```
480      * @param condition
481      *            Valeur de la clause WHERE de la requête SQL
482      * @return Liste des utilisateurs
483      */
484     private static ArrayList<JournalCon> getConnexionsList(int type,
485             String condition) {
486         ArrayList<JournalCon> connectList = new ArrayList<JournalCon>();
487         String where = (!condition.isEmpty()) ? " AND joco_action="
488                 + Utilities.formatSQL(condition) : "";
489 
490         String query = (type == 0) ? tbl_journal_con.getSelectConnexions(where)
491                 : tbl_journal_con.getSelectValidations(where);
492         ResultSet action = Sql.query(query);
493         try {
494             while (action.next()) {
495                 connectList.add(new JournalCon(action
496                         .getInt("PKNoJourConnection"), action
497                         .getString("joco_mail"),
498                         Utilities.formatSQLDate(action.getString("joco_date"),
499                                 "dd MMMM YY - HH:mm:ss"), action
500                                 .getString("joco_ip"), action
501                                 .getString("joco_os"), action
502                                 .getString("joco_browser"), action
503                                 .getString("joco_action"), action
504                                 .getString("joco_comment")));
505             }
506         } catch (SQLException e) {
507             e.printStackTrace();
508         }
509         return connectList;
510     }
511 
512     /**
513      * Récupération de la liste des actions journalisées pour les
514      * téléchargements
515      *
516      * @param condition
517      *            Valeur de la clause WHERE de la requête SQL
518      * @return Liste des téléchargements
519      */
520     private static ArrayList<JournalCon> getDownloadsList(String condition) {
521         ArrayList<JournalCon> downloadList = new ArrayList<JournalCon>();
522         String where = (!condition.isEmpty()) ? condition : "";
523 
524         String query = tbl_journal_fold.getSelectAll(where);
525         ResultSet action = Sql.query(query);
526 
527         try {
528             while (action.next()) {
529                 downloadList
530                         .add(new JournalCon(
531                                 action.getInt("PKNoJourFolder"),
532                                 action.getString("contact_name") + " <"
533                                         + action.getString("contact_mail")
534                                         + ">",
535                                 Utilities.formatSQLDate(
536                                         action.getString("joufo_date"),
537                                         "dd MMMM YY - HH:mm:ss"),
538                                 (action.getString("FKNoFile") == null) ? (action
539                                         .getString("joufo_action")
540                                         .equals("download")) ? Global
541                                         .i("CAPTION_ARCHIVE_OF_FOLDER") : "-"
```

```
                                        : action.getString("file_rename")
                                                + " ("
                                                + action.getString("file_name")
                                                + ")",
                                action.getString("user_mail"), // Nom de
                                                               // l'auteur
                                action.getString("folder_name"),
                                (action.getString("FKNoFile") == null && action
                                        .getString("joufo_action").equals(
                                                "download")) ? "archive"
                                        : action.getString("joufo_action"),
                                action.getString("FKNoFile")));
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return downloadList;
        }

        /**
         * Déselectionne tous les boutons du menu (par changement de style)
         */
        private static void deselectAllButtons() {
            for (Button button : btMenu) {
                button.removeStyleName(Runo.BUTTON_DEFAULT);
            }
        }
}
```



```
package modules.admin;

import java.sql.ResultSet;

/**
 * Module disponible uniquement par les administrateurs Il permet la
 * visualisation et la gestion des utilisateurs de l'interface.
 *
 * @author Dominique Roduit
 *
 */
public class AdminUsersModule {
    /**
     * Contient toutes les instances qui doivent être accessibles dans toute
     * l'application
     **/
    private static GlobalObjects global = CccvsTransfert.getGlobalMethod();
    /** Instance de la fenêtre principale **/
    private static Window mainWindow = CccvsTransfert.mainWindow;
    /** Layout Global de l'application **/
    private static MainLayout mainLayout = global.getMainLayout();
    /** Table contenant la liste des contacts **/
    private static Table tblUsers;
    /** Contient les entrées du tableau **/
    private static BeanItemContainer<User> cntUsers = new BeanItemContainer<User>(
            User.class);
    /** Action Utilisateur Simple **/
    private static Action ACTION_NO_ADMIN;
    /** Action Administrateur **/
    private static Action ACTION_ADMIN;
    /** Action Editer **/
    private static Action ACTION_EDIT;
    /** Action Supprimer **/
    private static Action ACTION_DELETE;
    /** Action valider **/
    private static Action ACTION_VALID;
    /** Action invalider **/
    private static Action ACTION_INVALID;

    /**
     * Chargement du bouton de retour et du titre dans le ruban du corps de la
     * page
     */
    public static HorizontalLayout getBodyRibbonContent() {

        ACTION_NO_ADMIN = new Action(Global.i("CAPTION_SIMPLE_USER"),
                new ThemeResource(Global.PATH_THEME_RESSOURCES
                        + "lock--minus.png"));
        ACTION_ADMIN = new Action(Global.i("CAPTION_ADMINISTRATOR"),
                new ThemeResource(Global.PATH_THEME_RESSOURCES + "unlock.png"));
        ACTION_EDIT = new Action(Global.i("CAPTION_EDIT"), new ThemeResource(
                Global.PATH_THEME_RESSOURCES + "edit.png"));
        ACTION_DELETE = new Action(Global.i("CAPTION_DELETE"),
                new ThemeResource(Global.PATH_THEME_RESSOURCES + "false.png"));
        ACTION_VALID = new Action(Global.i("CAPTION_VALID"), new ThemeResource(
                Global.PATH_THEME_RESSOURCES + "tick-shield.png"));
        ACTION_INVALID = new Action(
                Global.i("CAPTION_INVALID"),
                new ThemeResource(Global.PATH_THEME_RESSOURCES + "expanded.png"));

        System.out.println(Utilities.getCurrentDate()
                + " -- [Chargement du module] Ruban");
```

```
103
104         HorizontalLayout vlRibbon = new HorizontalLayout();
105         vlRibbon.setSizeFull();
106         vlRibbon.setSpacing(true);
107
108         Label lblTitle = new Label(Global.i("CAPTION_USERS"));
109         lblTitle.setStyleName(Runo.LABEL_H1);
110         vlRibbon.addComponent(lblTitle);
111         vlRibbon.setComponentAlignment(lblTitle, Alignment.MIDDLE_LEFT);
112
113         return vlRibbon;
114     }
115
116     /**
117      * Insertion du contenu dans le corps de la page.<br>
118      * Le contenu comprend le texte, extrait de la base de données ainsi qu'une
119      * image illustrative.
120      */
121     public static Table getBodyContent() {
122         System.out.println(Utilities.getCurrentDate()
123                 + " -- [Chargement du module] Body");
124
125         ArrayList<User> userList = getUserList();
126         cntUsers.removeAllItems();
127         cntUsers.addAll(userList);
128
129         tblUsers = new Table();
130         tblUsers.setSizeFull();
131         tblUsers.setSelectable(true);
132         tblUsers.setContainerDataSource(cntUsers);
133         tblUsers.setVisibleColumns(new String[] { "user_mail",
134                 "user_validation", "user_validation_date", "user_langue" });
135
136         // Génération de la colonne "Admin"
137         tblUsers.addGeneratedColumn("admin", new Table.ColumnGenerator() {
138             public Object generateCell(Table source, Object itemId,
139                     Object columnId) {
140                 User user = (User) itemId;
141                 return (user.isUser_admin()) ? new Embedded(null,
142                         new ThemeResource(Global.PATH_THEME_RESSOURCES
143                                 + "unlock.png")) : "";
144             }
145         });
146         tblUsers.setColumnWidth("admin", 40);
147         tblUsers.setColumnAlignment("admin", Table.ALIGN_CENTER);
148
149         // Génération de la colonne "Compte validé"
150         tblUsers.addGeneratedColumn("validation", new Table.ColumnGenerator() {
151             public Object generateCell(Table source, Object itemId,
152                     Object columnId) {
153                 User user = (User) itemId;
154                 return (user.getUser_validation()) ? new Embedded(null,
155                         new ThemeResource(Global.PATH_THEME_RESSOURCES
156                                 + "tick-shield.png")) : "";
157             }
158         });
159         tblUsers.setColumnWidth("validation", 45);
160         tblUsers.setColumnAlignment("validation", Table.ALIGN_CENTER);
161
162         // Génération de la colonne "Langue"
163         tblUsers.addGeneratedColumn("langue", new Table.ColumnGenerator() {
164             public Object generateCell(Table source, Object itemId,
```



```
165                     Object columnId) {
166                 User user = (User) itemId;
167                 return new Embedded(null, new ThemeResource(
168                         Global.PATH_THEME_RESSOURCES + user.getUser_langue()
169                                 + ".png"));
170             }
171         });
172         tblUsers.setColumnWidth("langue", 45);
173         tblUsers.setColumnAlignment("langue", Table.ALIGN_CENTER);
174
175         // Génération de la colonne "Interne"
176         tblUsers.addGeneratedColumn("interne", new Table.ColumnGenerator() {
177             public Object generateCell(Table source, Object itemId,
178                     Object columnId) {
179                 User user = (User) itemId;
180                 String imgInterne = (user.isInternal()) ? "home.png"
181                         : "icon-contact.png";
182                 return new Embedded(null, new ThemeResource(
183                         Global.PATH_THEME_RESSOURCES + imgInterne));
184             }
185         });
186         tblUsers.setColumnWidth("interne", 24);
187         tblUsers.setColumnAlignment("interne", Table.ALIGN_CENTER);
188
189         tblUsers.setColumnWidth("user_validation_date", 200);
190
191         tblUsers.setVisibleColumns(new String[] { "interne", "user_mail",
192                 "user_validation_date", "admin", "validation", "langue" });
193         tblUsers.setColumnHeaders(new String[] { "",
194                 Global.i("CAPTION_ADRESSE_EMAIL"),
195                 Global.i("CAPTION_VALIDATION_DATE"), "Admin",
196                 Global.i("CAPTION_VALIDATED"), Global.i("CAPTION_LANGUE") });
197
198         // Gestion du menu contextuel et clic droit
199         tblUsers.addActionHandler(new Action.Handler() {
200             public void handleAction(Action action, Object sender, Object target) {
201                 User user = (User) target;
202
203                 if (action == ACTION_ADMIN || action == ACTION_NO_ADMIN) {
204                     boolean newValue = !user.isUser_admin();
205                     Sql.exec(tbl_users.getUpdForAdminAssign(user.getPKNoUser(),
206                             newValue));
207
208                     Item item = tblUsers.getItem(target);
209                     item.getItemProperty("user_admin").setValue(newValue);
210                     tblUsers.refreshRowCache();
211                 }
212                 /*
213                  * if(action==ACTION_EDIT) {
214                  *
215                  * }
216                  */
217                 if (action == ACTION_DELETE) {
218                     System.out.println(user.getPKNoUser());
219                     Sql.exec(tbl_users.getDeleteUser(user.getPKNoUser()));
220                     tblUsers.removeItem(target);
221                 }
222                 if (action == ACTION_VALID || action == ACTION_INVALID) {
223                     boolean newValue = !user.getUser_validation();
224                     Sql.exec(tbl_users.getUpdForValidation(user.getPKNoUser(),
225                             newValue));
226
```

```
                    Item item = tblUsers.getItem(target);
                    item.getItemProperty("user_validation").setValue(newValue);
                    tblUsers.refreshRowCache();
                }
            }

            public Action[] getActions(Object target, Object sender) {
                if (target != null) {
                    User user = (User) target;
                    Action[] actions = new Action[3];
                    actions[0] = (!user.getUser_validation()) ? ACTION_VALID
                            : ACTION_INVALID;
                    actions[1] = (user.isUser_admin()) ? ACTION_NO_ADMIN
                            : ACTION_ADMIN;
                    // actions[2] = ACTION_EDIT;
                    actions[2] = ACTION_DELETE;
                    return actions;
                } else {
                    return null;
                }
            }
        });
        tblUsers.addListener(new ItemClickListener() {
            public void itemClick(ItemClickEvent event) {
                if (event.getButton() == ItemClickEvent.BUTTON_RIGHT) {
                    tblUsers.setValue(null); // Dé-sélectionne tout
                    tblUsers.select(event.getItemId()); // Sélectionne la ligne
                                                        // en cours
                }
            }
        });

        return tblUsers;
    }

    /**
     * Mise à jour de la table des utilisateurs
     */
    private static void reloadTableUsers() {
        cntUsers.removeAllItems();
        cntUsers.addAll(getUserList());
    }

    /**
     * Récupération de la liste des utilisateurs
     *
     * @return Liste des utilisateurs
     */
    private static ArrayList<User> getUserList() {
        ArrayList<User> userList = new ArrayList<User>();
        ResultSet user = Sql.query(tbl_users.getSelectAll());
        try {
            while (user.next()) {
                userList.add(new User(user.getInt("PKNoUser"), user
                        .getString("user_mail"), user.getString("user_pass"),
                        user.getBoolean("user_internal"), user
                                .getBoolean("user_validation"), user
                                .getString("user_validation_date"), user
                                .getString("user_langue"), user
                                .getBoolean("user_admin")));
            }
        } catch (SQLException e) {
```



```
            e.printStackTrace();
        }
        return userList;
    }
}
```

PACKAGE

SQL.QUERY

- tbl_contacts
- tbl_files
- tbl_folders
- tbl_journal_con
- tbl_journal_fold
- tbl_recipients
- tbl_users

Classes qui répertorient les requêtes SQL utilisées. Elles ne sont majoritairement pas exécutées mais uniquement retournées sous forme de chaine de caractères.

```java
package sql.query;

import java.sql.ResultSet;

/**
 * Cette class contient toutes les requêtes SQL utiles à la gestion de la table
 * des contacts.
 *
 * @author Dominique Roduit
 *
 */
public class tbl_contacts {
    /** Nom de la table sur laquelle on effectue des opérations dans cette class
  **/
    private static final String TABLE_NAME = "trans_contacts";

    /**
     * Requête pour la sélection de la liste des contacts
     *
     * @return (String) Requête SQL Formatée
     */
    public static String getSelectAllContactQuery() {
        return "SELECT * FROM " + TABLE_NAME + " WHERE FKNoUser="
                + Utilities.formatSQL(UserSession.getID());
    }

    /**
     * Requête pour la sélection d'un contact par sa clé primaire
     *
     * @param PK
     *            (String) Clé primaire du contact
     * @return (String) Requête SQL Formatée
     */
    public static String getSelectByPKContactQuery(String PK) {
        return "SELECT * FROM " + TABLE_NAME + " WHERE PKNoContact=" + PK;
    }

    /**
     * Requête pour la suppression d'un contact par sa clé primaire
     *
     * @param PK
     *            (String) Clé primaire du contact
     * @return (String) Requête SQL Formatée
     */
    public static String getDeleteContactQuery(String PK) {
        return "DELETE FROM " + TABLE_NAME + " WHERE PKNoContact="
                + Utilities.formatSQL(PK);
    }

    /**
     * Requête pour l'insertion d'un contact
     *
     * @param name
     *            (String) Nom du contact
     * @param email
     *            (String) E-mail du contact
     * @return (String) Requête SQL Formatée
     */
    public static String getInsertContactQuery(String name, String email) {
        return "INSERT INTO " + TABLE_NAME + " VALUES " + "(NULL, "
                + Utilities.formatSQL(UserSession.getID()) + ", "
                + Utilities.formatSQL(name) + ", " + Utilities.formatSQL(email)
```



```java
                + ", now(), "
                + Utilities.formatSQL(Utilities.isInternal(email)) + ")";
    }

    /**
     * Requête pour la mise à jour d'un contact
     *
     * @param PK
     *            (String) Clé primaire du contact
     * @param name
     *            (String) Nom du contact
     * @param email
     *            (String) E-mail du contact
     * @return (String) Requête SQL Formatée
     */
    public String getUpdateContactQuery(String PK, String name, String email) {
        return "UPDATE " + TABLE_NAME + " SET " + "contact_name="
                + Utilities.formatSQL(name) + ", " + "contact_mail="
                + Utilities.formatSQL(email) + ", " + "contact_internal="
                + Utilities.formatSQL(Utilities.isInternal(email)) + ", "
                + "contact_creation_date=now() " + "WHERE FKNoUser="
                + Utilities.formatSQL(UserSession.getID())
                + " AND PKNoContact=" + Utilities.formatSQL(PK);
    }

    /**
     * Retourne l'ID du dernier contact
     *
     * @return ID du dernier contact
     */
    public static String getLastContact() {
        return "SELECT MAX(PKNoContact) AS LAST FROM " + TABLE_NAME
                + " WHERE FKNoUser=" + Utilities.formatSQL(UserSession.getID());
    }

    /**
     * Retourne le nombre de contact dans la liste d'un utilisateur
     *
     * @return Requête SQL
     */
    public static int getNumberOfContact() {
        ResultSet data = Sql.query("SELECT COUNT(*) AS NUMBER FROM "
                + TABLE_NAME + " WHERE FKNoUser="
                + Utilities.formatSQL(UserSession.getID()));
        int number = 0;
        try {
            if (data.first()) {
                number = data.getInt("NUMBER");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return number;
    }

    /**
     * Sélection d'un contact en fonction d'un paramètre crypté passé dans l'URL
     *
     * @param URLParameter
     *            Valeur du paramètre de l'URL
     * @return Requête SQL
     */
```

```
130     public static String getContactByCryptedURL(String URLParameter) {
131         return "SELECT * FROM " + TABLE_NAME + " WHERE SHA2(CONCAT('"
132                 + Global.getParm("SECRET_KEY") + "', PKNoContact), 256)="
133                 + Utilities.formatSQL(URLParameter) + "";
134     }
135 
136     /**
137      * Sélection des contacts attribués comme destinataires d'un dossier
138      *
139      * @param PKNoFolder
140      *            Clé primaire du dossier
141      * @return Requête SQL
142      */
143     public static String getSelectContactsFromFolder(int PKNoFolder) {
144         return "SELECT * FROM  trans_recipients LEFT JOIN " + TABLE_NAME
145                 + " ON PKNoContact=FKNoContact WHERE FKNoFolder="
146                 + Utilities.formatSQL(PKNoFolder)
147                 + " ORDER BY contact_name ASC, contact_mail ASC";
148     }
149 }
150 
```



```
1 package sql.query;
2 
3 import java.sql.ResultSet;
9 
10 /**
11  * Contient toutes les requêtes effectuées sur la table <b>trans_files</b>
12  *
13  * @author Dominique Roduit
14  *
15  */
16 public class tbl_files {
17     /** Nom de la table sur laquelle on effectue des opérations dans cette class **/
18     private static final String TABLE_NAME = "trans_files";
19 
20     /**
21      * Retourne tous les fichiers d'un dossier
22      *
23      * @param FKNoFolder
24      *            La clé primaire du dossier contenant les fichiers
25      * @return Requête SQL
26      */
27     public static String getSelectAllFilesFromFolder(int FKNoFolder) {
28         return "SELECT * FROM " + TABLE_NAME + " WHERE FKNoFolder="
29                 + Utilities.formatSQL(FKNoFolder);
30     }
31 
32     /**
33      * Retourne le nombres de fichiers contenus dans un dossier
34      *
35      * @param FKNoFolder
36      *            Clé Primaire du dossier
37      * @return Nombre de fichiers du dossier
38      */
39     public static int getFileNumberFromFolder(int PKNoFolder) {
40         ResultSet data = Sql.query("SELECT COUNT(*) AS NUMBER FROM "
41                 + TABLE_NAME + " WHERE FKNoFolder="
42                 + Utilities.formatSQL(PKNoFolder));
43         int number = 0;
44         try {
45             if (data.first()) {
46                 number = data.getInt("NUMBER");
47             }
48         } catch (SQLException e) {
49             e.printStackTrace();
50         }
51         return number;
52     }
53 
54     /**
55      * Retourne la taille total des fichiers contenus dans un dossier
56      *
57      * @param FKNoFolder
58      *            Clé Primaire du dossier
59      * @return Taille total des fichiers contenu dans le dossier (en octet)
60      */
61     public static int getFileSizeFromFolder(int PKNoFolder) {
62         ResultSet data = Sql.query("SELECT SUM(file_size) AS SIZE FROM "
63                 + TABLE_NAME + " WHERE FKNoFolder="
64                 + Utilities.formatSQL(PKNoFolder));
65         int size = 0;
66         try {
67             if (data.first()) {
```

```
68                 size = data.getInt("SIZE");
69             }
70         } catch (SQLException e) {
71             e.printStackTrace();
72         }
73         return size;
74     }
75 }
76
```



```
1 package sql.query;
2
3 import java.sql.ResultSet;
10
11 /**
12  * Contient toutes les requêtes effectuées sur la table <b>trans_folders</b>
13  *
14  * @author Dominique Roduit
15  *
16  */
17 public class tbl_folders {
18     /** Nom de la table sur laquelle on effectue des opérations dans cette class
   **/
19     private static final String TABLE_NAME = "trans_folders";
20
21     /**
22      * Requête pour la sélection de la liste des contacts
23      *
24      * @return (String) Requête SQL Formatée
25      */
26     public static String getAllFolders() {
27         return "SELECT * FROM " + TABLE_NAME + " WHERE FKNoUser="
28                 + Utilities.formatSQL(UserSession.getID())
29                 + " ORDER BY folder_archive, folder_creation_date DESC";
30     }
31
32     /**
33      * Requête pour la suppression d'un dossier
34      *
35      * @param PK
36      *             La clé primaire du dossier à supprimer
37      * @return Requête SQL Formatée
38      */
39     public static String getDeleteFolder(String PK) {
40         return "DELETE FROM " + TABLE_NAME + " WHERE PKNoFolder="
41                 + Utilities.formatSQL(PK);
42     }
43
44     /**
45      * Requête pour l'insertion d'un dossier
46      *
47      * @param name
48      *             Nom du dossier
49      * @param expiration
50      *             Nombre de jour ou nous mettons a disposition le dossier
51      * @return Requête SQL
52      */
53     public static String getInsertFolder(String name, double expiration,
54             String description) {
55         int expirationInt = (int) expiration;
56         return "INSERT INTO " + TABLE_NAME + " VALUES(NULL, "
57                 + Utilities.formatSQL(UserSession.getID()) + ", "
58                 + Utilities.formatSQL(name) + ", "
59                 + Utilities.formatSQL(description)
60                 + ", now(), DATE_ADD(NOW(), INTERVAL " + expirationInt
61                 + " DAY), 0)";
62     }
63
64     /**
65      * Requête pour la récupération du dernier dossier créé par l'utilisateur
66      *
67      * @return La PK du dernier dossier créé par l'utilisateur
```

```
68      */
69     public static String getSelectMaxPK() {
70         return "SELECT MAX(PKNoFolder) AS PKNoFolder FROM " + TABLE_NAME
71                 + " WHERE FKNoUser=" + Utilities.formatSQL(UserSession.getID());
72     }
73
74     /**
75      * Retourne les informations sur un dossier
76      *
77      * @param pKNoFolder
78      *             Clé primaire du dossier
79      * @return Requête SQL
80      */
81     public static String getSelectFolderInfos(int pKNoFolder) {
82         return "SELECT * FROM " + TABLE_NAME + " WHERE PKNoFolder="
83                 + Utilities.formatSQL(pKNoFolder);
84     }
85
86     /**
87      * Retourne le nombre de dossiers d'un utilisateur
88      *
89      * @return Requête SQL
90      */
91     public static int getNumberOfFolders() {
92         ResultSet data = Sql.query("SELECT COUNT(*) AS NUMBER FROM "
93                 + TABLE_NAME + " WHERE FKNoUser="
94                 + Utilities.formatSQL(UserSession.getID()));
95         int number = 0;
96         try {
97             if (data.first()) {
98                 number = data.getInt("NUMBER");
99             }
100        } catch (SQLException e) {
101            e.printStackTrace();
102        }
103        return number;
104    }
105
106    /**
107     * Retourne le nombre de jour ou le dossier est disponible (Date
108     * d'expiration-Date de création)
109     *
110     * @param PKNoFolder
111     *             Clé primaire du dossier
112     * @return Nombre de jour de disponibilité du dossier
113     */
114    public static int getNumberOfDayFolderIsAvailable(int PKNoFolder) {
115        ResultSet data = Sql
116                .query("SELECT DATEDIFF(folder_expiration,folder_creation_date) AS
    NUMBER FROM "
117                        + TABLE_NAME
118                        + " WHERE PKNoFolder="
119                        + Utilities.formatSQL(PKNoFolder));
120        int number = 0;
121        try {
122            if (data.first()) {
123                number = data.getInt("NUMBER");
124            }
125        } catch (SQLException e) {
126            e.printStackTrace();
127        }
128        return number;
```



```
129    }
130
131    /**
132     * Retourne le nombre de jours restant pour un dossier avant son expiration
133     *
134     * @param PKNoFolder
135     *             Clé primaire du dossier
136     * @return Durée de vie restante du dossier
137     */
138    public static int getRemainingDaysFolderIsAvailable(int PKNoFolder) {
139        ResultSet data = Sql
140                .query("SELECT DATEDIFF(folder_expiration,NOW()) AS NUMBER FROM "
141                        + TABLE_NAME
142                        + " WHERE PKNoFolder="
143                        + Utilities.formatSQL(PKNoFolder));
144        int number = 0;
145        try {
146            if (data.first()) {
147                number = data.getInt("NUMBER");
148            }
149        } catch (SQLException e) {
150            e.printStackTrace();
151        }
152        return number;
153    }
154
155    /**
156     * Retourne le nombde d'heures restantes avant l'expiration d'un dossier
157     *
158     * @param PKNoFolder
159     *             Clé primaire du dossier
160     * @return Heures restantes avant la destruction du dossier
161     */
162    public static int getRemainingHoursFolderIsAvailable(int PKNoFolder) {
163        ResultSet data = Sql
164                .query("SELECT ABS(TIMESTAMPDIFF( HOUR , folder_expiration, NOW( )
    )) AS NUMBER FROM "
165                        + TABLE_NAME
166                        + " WHERE PKNoFolder="
167                        + Utilities.formatSQL(PKNoFolder));
168        int number = 0;
169        try {
170            if (data.first()) {
171                number = data.getInt("NUMBER");
172            }
173        } catch (SQLException e) {
174            e.printStackTrace();
175        }
176        return number;
177    }
178
179    /**
180     * Sélection d'un dossier en fonction d'un paramètre crypté passé dans l'URL
181     *
182     * @param URLParameter
183     *             Valeur du paramètre de l'URL
184     * @return Requête SQL
185     */
186    public static String getFolderByCryptedURL(String URLParameter) {
187        return "SELECT * FROM " + TABLE_NAME + " WHERE SHA2(CONCAT('"
188                + Global.getParm("SECRET_KEY") + "', PKNoFolder), 256)="
189                + Utilities.formatSQL(URLParameter) + "";
```

```
190     }
191
192 }
193
```



```
1 package sql.query;
2
3 import java.sql.ResultSet;
9
10 /**
11  * JOURNALISATION<br>
12  * Contient toutes les requêtes effectuées sur la table
13  * <b>trans_journal_connections</b>
14  *
15  * @author Dominique Roduit
16  *
17  */
18 public class tbl_journal_con {
19     /** Nom de la table sur laquelle on effectue des opérations dans cette class **/
20     private static final String TABLE_NAME = "trans_journal_connections";
21
22     /**
23      * Insertion d'une entrée dans le journal des connexions
24      *
25      * @param email
26      *            E-mail de l'utilisateur
27      * @param action
28      *            Action effectuée / Evenement
29      * @param comment
30      *            Commentaire sur l'action effectuée
31      * @return Requête SQL formatée
32      */
33     public static String getInsertQuery(String email, String action,
34             String comment) {
35         String ip = (Global.browser.getAddress() != null) ? Global.browser
36                 .getAddress() : "";
37         String os = (Utilities.getOS() != null) ? Utilities.getOS() : "";
38         String browser = (Utilities.getBrowserAndVersion() != null) ? Utilities
39                 .getBrowserAndVersion() : "";
40
41         return "INSERT INTO " + TABLE_NAME + " VALUES(NULL, "
42                 + Utilities.formatSQL(email) + ", now(), "
43                 + Utilities.formatSQL(ip) + ", " + Utilities.formatSQL(os)
44                 + ", " + Utilities.formatSQL(browser) + ", "
45                 + Utilities.formatSQL(action) + ", "
46                 + Utilities.formatSQL(comment) + ")";
47     }
48
49     /**
50      * Retourne le nombres de fois qu'un utilisateur a tenté de se connecter
51      * sans valider son compte
52      *
53      * @param email
54      *            Adresse e-mail de l'utilisateur
55      * @return Nombre de connection sans avoir validé le compte
56      */
57     public static int getNumberOfConnexionsWhithoutValidation(String email) {
58         ResultSet data = Sql.query("SELECT COUNT(*) AS NUMBER FROM "
59                 + TABLE_NAME + " WHERE joco_mail=" + Utilities.formatSQL(email)
60                 + " AND joco_action='con_no_validate'");
61         int number = 0;
62         try {
63             if (data.first()) {
64                 number = data.getInt("NUMBER");
65             }
66         } catch (SQLException e) {
67             e.printStackTrace();
```

```
68          }
69          return number;
70      }
71
72      /**
73       * Requête pour la sélection des journalisations sur les connexions
74       *
75       * @return Requête SQL
76       */
77      public static String getSelectConnexions(String where) {
78          return "SELECT * FROM " + TABLE_NAME
79                  + " WHERE joco_action LIKE 'con_%' " + where
80                  + " ORDER BY joco_date DESC LIMIT 100";
81      }
82
83      /**
84       * Requête pour la sélection des journalisations sur les validations des
85       * comptes
86       *
87       * @return Requête SQL
88       */
89      public static String getSelectValidations(String where) {
90          return "SELECT * FROM " + TABLE_NAME
91                  + " WHERE joco_action LIKE 'validation_%' " + where
92                  + " ORDER BY joco_date DESC LIMIT 100";
93      }
94
95  }
96
```



```
1  package sql.query;
2
3  import java.sql.ResultSet;
9
10 /**
11  * JOURNALISATION<br>
12  * Contient toutes les requêtes effectuées sur la table
13  * <b>trans_journal_download</b>
14  *
15  * @author Dominique Roduit
16  *
17  */
18 public class tbl_journal_fold {
19      /** Nom de la table sur laquelle on effectue des opérations dans cette class **/
20      private static final String TABLE_NAME = "trans_journal_folders";
21
22      /**
23       * Insertion d'une entrée dans le journal des téléchargements
24       *
25       * @param FKNoContact
26       *            Clé étrangère vers le contact
27       * @param FKNoFolder
28       *            Clé étrangère vers le dossier
29       * @param FKNoFile
30       *            Clé étrangère vers le fichier
31       * @return Requête SQL formatée
32       */
33      public static String getInsertQuery(String action, int FKNoContact,
34              int FKNoFolder, int FKNoFile) {
35          String file = (FKNoFile == 0) ? "NULL" : Utilities.formatSQL(FKNoFile);
36          return "INSERT INTO " + TABLE_NAME + " VALUES(NULL, "
37                  + Utilities.formatSQL(FKNoContact) + ", "
38                  + Utilities.formatSQL(FKNoFolder) + ", " + file + ", "
39                  + Utilities.formatSQL(action) + ", now())";
40      }
41
42      /**
43       * Retourne la requête de sélection des informations journalisées pour un
44       * contact précisé
45       *
46       * @param pKNoContact
47       *            Clé primaire du contact
48       * @return Requête SQL
49       */
50      public static String getSelectAllForContact(int pKNoContact) {
51          return "SELECT * FROM " + TABLE_NAME + " "
52                  + "LEFT JOIN trans_files ON PKNoFile=FKNoFile "
53                  + "WHERE FKNoContact=" + Utilities.formatSQL(pKNoContact);
54      }
55
56      /**
57       * Retourne la requête de sélection des informations journalisées sur un
58       * dossier, pour un contact précisé
59       *
60       * @param pKNoContact
61       *            Clé primaire du contact
62       * @param PKNoFolder
63       *            Clé primaire du dossier
64       * @return Requête SQL
65       */
66      public static String getSelectAllForContactAndFolder(int pKNoContact,
67              int PKNoFolder) {
```

```
68          return "SELECT * FROM " + TABLE_NAME + " "
69                  + "LEFT JOIN trans_files ON PKNoFile=FKNoFile "
70                  + "WHERE FKNoContact=" + Utilities.formatSQL(pKNoContact)
71                  + " AND " + TABLE_NAME + ".FKNoFolder="
72                  + Utilities.formatSQL(PKNoFolder);
73      }
74
75      /**
76       * Requête de sélection de toutes les informations journalisées sur les
77       * dossiers
78       *
79       * @param where
80       *            Clause WHERE de la requête
81       * @return Requête SQL
82       */
83      public static String getSelectAll(String where) {
84          String Where = (!where.isEmpty()) ? " WHERE " + where + " " : "";
85          return "SELECT * FROM " + TABLE_NAME + " "
86                  + "LEFT JOIN trans_contacts ON PKNoContact=FKNoContact "
87                  + "LEFT JOIN trans_files ON PKNoFile=FKNoFile "
88                  + "LEFT JOIN trans_folders ON PKNoFolder=" + TABLE_NAME
89                  + ".FKNoFolder "
90                  + "LEFT JOIN trans_users ON trans_folders.FKNoUser = PKNoUser "
91                  + Where + "LIMIT 100";
92      }
93
94  }
95
```



```
1  package sql.query;
2
3  import toolbox.Utilities;
4
5  /**
6   * Contient toutes les requêtes effectuées sur la table <b>trans_recipients</b>
7   *
8   * @author Dominique Roduit
9   *
10  */
11 public class tbl_recipients {
12      /** Nom de la table sur laquelle on effectue des opérations dans cette class **/
13      private static final String TABLE_NAME = "trans_recipients";
14
15      /** Requête d'insertion pour les destinataires **/
16      public static String getInsertRecipient(int FKNoFolder, int FKNoContact) {
17          return "INSERT INTO " + TABLE_NAME + " VALUES (NULL, "
18                  + Utilities.formatSQL(FKNoFolder) + ", "
19                  + Utilities.formatSQL(FKNoContact) + ")";
20      }
21
22      /**
23       * Requête de sélection des destinataire pour un dossier
24       *
25       * @param pKNoFolder
26       *             PK du dossier
27       * @return Requête SQL
28       */
29      public static String getSelectRecipientsFromFolder(int pKNoFolder) {
30          return "SELECT * FROM " + TABLE_NAME + " "
31                  + "LEFT JOIN trans_contacts ON FKNoContact=PKNoContact "
32                  + "WHERE FKNoFolder=" + Utilities.formatSQL(pKNoFolder);
33      }
34 }
35
```

```java
package sql.query;

import global.Global;

/**
 * Contient toutes les requêtes effectuées sur la table <b>trans_users</b>
 *
 * @author Dominique Roduit
 *
 */
public class tbl_users {
    /** Nom de la table sur laquelle on effectue des opérations dans cette class **/
    private static final String TABLE_NAME = "trans_users";

    /**
     * Sélection d'un utilisateur par son adresse e-mail
     *
     * @param email
     *            Adresse de l'utilisateur
     * @return Requête SQL
     */
    public static String getUserByEmail(String email) {
        return "SELECT * FROM " + TABLE_NAME + " WHERE user_mail="
                + Utilities.formatSQL(email);
    }

    /**
     * Mise à jour du champs de validation du compte utilisateur
     *
     * @param email
     *            Adresse de l'utilisateur qui valide son compte
     * @return Requête SQL
     */
    public static String getUpdByMailValidation(String email) {
        return "UPDATE "
                + TABLE_NAME
                + " SET user_validation=1, user_validation_date=now() WHERE user_mail="
                + Utilities.formatSQL(email);
    }

    /**
     * Sélection d'un utilisateur en fonction d'un paramètre crypté passé dans
     * l'URL
     *
     * @param URLParameter
     *            Valeur du paramètre de l'URL
     * @return Requête SQL
     */
    public static String getUserByCryptedURL(String URLParameter) {
        return "SELECT * FROM " + TABLE_NAME + " WHERE SHA2(CONCAT('"
                + Global.getParm("SECRET_KEY") + "', user_mail), 256)="
                + Utilities.formatSQL(URLParameter) + "";
    }

    /**
     * Retourne la requête d'insertion d'un utilisateur
     *
     * @param email
     *            Adresse e-mail
     * @param pass
     *            Mot de passe
     * @param internal
     *            L'utilisateur est un interne ou non
     * @return Requête SQL
     */
    public static String getInsertQuery(String email, String pass,
            boolean isInternal) {
        // Par défaut, le compte n'est pas validé
        boolean account_valid = false;

        return "INSERT INTO "
                + TABLE_NAME
                + " VALUES(NULL, "
                + Utilities.formatSQL(email)
                + ", "
                + Utilities.formatSQL(pass)
                + ","
                + Utilities.formatSQL(isInternal)
                + ", "
                + Utilities.formatSQL(account_valid)
                + ", now(), "
                + Utilities.formatSQL(Global.browser.getLocale().toString()
                        .substring(0, 2)) + ", 0)";
    }

    /**
     * Requête pour la mise à jour de la langue de l'utilisateur
     *
     * @param newLanguage
     *            Langue choisie par l'utilisateur
     * @return Requête SQL
     */
    public static String getUpdLanguage(String newLanguage) {
        return "UPDATE " + TABLE_NAME + " SET user_langue="
                + Utilities.formatSQL(newLanguage) + " WHERE PKNoUser="
                + UserSession.getID();
    }

    /**
     * Requête pour la sélection des informations sur un utilisateur
     *
     * @param PKNoUser
     *            Clé primaire de l'utilisateur
     * @return Requête SQL
     */
    public static String getSelectUserInfos(int PKNoUser) {
        return "SELECT * FROM " + TABLE_NAME + " WHERE PKNoUser="
                + Utilities.formatSQL(PKNoUser);
    }

    /**
     * Sélection de tous les contacts de l'interface utilisateur
     *
     * @return Requête SQL
     */
    public static String getSelectAll() {
        return "SELECT * FROM " + TABLE_NAME
                + " ORDER BY user_internal ASC, PKNoUser";
    }

    /**
     * Requête de mise à jour d'un utilisateur (Admin ou pas)
```

```
125      *
126      * @param newValue
127      *            true si admin
128      * @return Requête SQL
129      */
130     public static String getUpdForAdminAssign(int PKNoUser, boolean newValue) {
131         return "UPDATE " + TABLE_NAME + " SET user_admin ="
132                 + Utilities.formatSQL(newValue) + " WHERE PKNoUser="
133                 + Utilities.formatSQL(PKNoUser);
134     }
135
136     /**
137      * Requête de validation d'un compte utilisateur
138      *
139      * @param newValue
140      *            true pour valider
141      * @return Requête SQL
142      */
143     public static String getUpdForValidation(int PKNoUser, boolean newValue) {
144         return "UPDATE " + TABLE_NAME + " SET user_validation ="
145                 + Utilities.formatSQL(newValue) + " WHERE PKNoUser="
146                 + Utilities.formatSQL(PKNoUser);
147     }
148
149     /**
150      * Requête de suppression d'un utilisateur
151      *
152      * @param pkNoUser
153      *            Clé primaire de l'utilisateur
154      * @return Requête SQL
155      */
156     public static String getDeleteUser(int pkNoUser) {
157         return "DELETE FROM " + TABLE_NAME + " WHERE PKNoUser="
158                 + Utilities.formatSQL(pkNoUser);
159     }
160 }
161
```

PACKAGE

TOOLBOX

- Mailer
- SHA256
- Sql
- Utilities
- ZipFileWritter

Regroupement de classes essentielles utilisées fréquemment sur l'ensemble de l'application.

```java
package toolbox;

import global.Global;

/**
 * Class qui répertorie différents outils relatifs à l'utilisation des adresses
 * e-mails<br>
 * <b>Outils principaux :</b><br>
 * <ul>
 * <li>Envoi de mails en java via le protocole SMPT (requis : librairie
 * javamail).</li>
 * <li>Validation et contrôles de formats d'adresses</li>
 * </ul>
 *
 * @author Dominique Roduit
 *
 */
public class Mailer {
    /**
     * Nom ou adresse ip du serveur SMTP
     */
    private final static String SERVER = "localhost";
    /**
     * Suffixe d'une addresse e-mail standard de la CCCVs
     */
    private final static String SUFFIXE_FORMAT = Global
            .getParm("EMAIL_INTERNE_SUFFIXE");
    /**
     * Adresse mail utilisée pour envoyer des messages automatiques aux
     * utilisateurs
     */
    public final static String APP_MAIL_FROM = Global.getParm("APP_MAIL_FROM");
    /**
     * Nom d'affichage de l'adresse e-mail définie par APP_MAIL_FROM
     */
    public final static String APP_MAIL_FROM_NAME = Global
            .getParm("APP_MAIL_FROM_NAME");

    /**
     * Envoi d'un mail au(x) destinataire(s) spécifié(s).
     *
     * @param from
     *            (String) Auteur du mail
     * @param from_name
     *            (String) Nom de l'auteur du mail
     * @param to
     *            (String) Destinataires (séparés par des virgules)
     * @param subject
     *            (String) Sujet du message
     * @param content
     *            (String) Contenu du mail au format HTML
     */
    public static void send(String from, String from_name, String to,
            String subject, String content) {
        try {
            Properties prop = System.getProperties();
            prop.put("mail.smtp.host", SERVER);

            Session session = Session.getInstance(prop);
            Message message = new MimeMessage(session);

            // Auteur
```



```java
            message.setFrom(new InternetAddress(from, from_name));

            // Destinataire(s)
            InternetAddress[] internetAddresses;
            if (to.indexOf(",") > -1) {
                String[] to_array = to.split(",");
                internetAddresses = new InternetAddress[to_array.length + 1];
                for (int i = 0; i < to_array.length; i++) {
                    internetAddresses[i] = new InternetAddress(to_array[i]);
                }
            } else {
                internetAddresses = new InternetAddress[1];
                internetAddresses[0] = new InternetAddress(to);
            }
            message.setRecipients(Message.RecipientType.TO, internetAddresses);

            // Sujet
            subject = (Global.isDev) ? subject : new String(subject.getBytes(),
                    "UTF-8");
            message.setSubject(subject);

            // Contenu du mail
            content = (Global.isDev) ? content : new String(content.getBytes(),
                    "UTF-8");
            message.setContent(content, "text/html; charset=UTF-8");

            message.setSentDate(new Date());
            session.setDebug(true);

            Transport.send(message);
        } catch (NoSuchProviderException e) {
            System.err.println("Pas de transport disponible pour ce protocole");
            System.err.println(e);
        } catch (AddressException e) {
            System.err.println("Adresse invalide");
            System.err.println(e);
        } catch (MessagingException e) {
            System.err.println("Erreur dans le message");
            System.err.println(e);
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }

    /**
     * Envoi d'un mail automatique, l'adresse de l'auteur reste fixe et le nom
     * aussi
     *
     * @param to
     *            Destinataires séparés par des virgules
     * @param subject
     *            Sujet du message
     * @param content
     *            Contenu du mail
     */
    public static void send(String to, String subject, String content) {
        send(APP_MAIL_FROM, APP_MAIL_FROM_NAME, to, subject, content);
    }

    /**
     * Cette classe regroupe des outils très utiles pouvant être appliqués aux
     * adresses mails. Elle peut être instanciée comme suit :
```

```java
     * Mailer.EmailValidator emailValidator = new Mailer.EmailValidator();
     *
     * @author Dominique Roduit
     */
    public static class EmailValidator {
        /**
         * Validation du format d'une adresse e-mail
         *
         * @param mailAddresse
         *            Adresse e-mail à valider
         * @return true : Adresse valide<br>
         *         false : Adresse invalide
         */
        public static boolean validate(final String mailAddresse) {
            String EMAIL_PATTERN = "^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
                    + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$";
            Pattern pattern = Pattern.compile(EMAIL_PATTERN);
            Matcher matcher = pattern.matcher(mailAddresse);
            return matcher.matches();
        }

        /**
         * Contrôle que la personne soit interne (de la cccvs) ou non
         *
         * @return true : personne interne (de la cccvs)<br>
         *         false : personne externe
         */
        public static boolean is_internal(String email) {
            return (email.substring(email.indexOf("@")).equals(SUFFIXE_FORMAT)) ?
true
                    : false;
        }
    }

    /**
     * Envoi du mail contenant le lien de validation du compte utilisateur
     *
     * @param to
     *            Adresse e-mail de destination
     */
    public static void sendAccountValidationMail(String to) {
        // Envoi d'un mail contenant un lien de validation crypté
        String parmRestart = (Global
                .getParm("URL_RESTART_APPLICATION_VALIDATION") == "1") ?
"?restartApplication&"
                : "?";

        // Lien crypté
        String cryptedLink = "<a href=\"" + Global.URL + parmRestart
                + Global.getParm("URL_PARM_VALIDATION") + "="
                + SHA256.getHashValue(Global.getParm("SECRET_KEY") + to)
                + "\">" + Global.i("CAPTION_VALIDER_INSCRIPTION") + "</a>";

        // Contenu du mail
        String mailContent = Global.getParm("MAIL_HEADER_CONTENT")
                + Global.getParm("MAIL_VALIDATION_CONTENT") + cryptedLink
                + Global.getParm("MAIL_FOOTER_CONTENT");

        Mailer.send(to, Global.i("CAPTION_ACCOUNT_VALIDATION"), mailContent);
    }

    /**
```



```java
     * Envoi du mail contenant le lien de téléchargement d'un dossier. Il peut
     * être envoyé à un ou plusieurs destinataires.
     *
     * @param to
     *            Un ou plusieurs destinataires (ex. dominique@roduit.com,
     *            dominique.roduit@avs.vs.ch)
     * @param PKNoFolder
     *            Clé primaire du dossier envoyé
     * @param PKNoContact
     *            Clé primaire du contact à qui le dossier est envoyé
     * @param expiration_date
     *            Date d'expiration du dossier
     * @param expiration
     *            Durée de vie du dossier en jours
     */
    public static void sendDownloadLink(String to, int PKNoFolder,
            int PKNoContact, String expiration_date, int expiration,
            String description, String folder_name) {
        // Lien crypté
        String cryptedLink = "<a href=\""
                + Global.URL
                + "?restartApplication&"
                + Global.getParm("URL_PARM_DOWNLOAD")
                + "="
                + SHA256.getHashValue(Global.getParm("SECRET_KEY") + PKNoFolder)
                + "&idm="
                + SHA256.getHashValue(Global.getParm("SECRET_KEY") + to)
                + "&id="
                + SHA256.getHashValue(Global.getParm("SECRET_KEY")
                        + PKNoContact) + "\">"
                + Global.i("CAPTION_DOWNLOAD_FILES") + "</a>";

        String desc = (description != null) ? (description.length() > 0) ?
description
                + "<br><br>"
                : ""
                : "";

        // Contenu du mail
        String mailContent = Global.getParm("MAIL_HEADER_CONTENT")
                + Global.getParm("MAIL_DOWNLOAD_CONTENT")
                        .replace("%folder_name%", folder_name)
                        .replace("%author%", UserSession.getMail())
                        .replace("%expiration%", Integer.toString(expiration))
                        .replace(
                                "%expiration_date%",
                                Utilities.formatSQLDate(expiration_date,
                                        "dd.MM.YY"))
                        .replace(
                                "%expiration_heure%",
                                Utilities.zeroFill(Integer.parseInt(Utilities
                                        .formatSQLDate(expiration_date, "HH")) + 1)
                                        + ":00").replace("%description%", desc)
                + cryptedLink + Global.getParm("MAIL_FOOTER_CONTENT");

        Mailer.send(to, Global.i("SUBJECT_MAIL_DOWNLOAD_LINK"), mailContent);
    }
}
```

```java
package toolbox;

import java.security.MessageDigest;

/**
 * Class de Hachage de valeurs.<br>
 * Cette class permet le hachage de valeur en utilisant l'algorithme de cryptage
 * <b>SHA-256</b>.<br>
 * Le hachage est particulièrement utilisé pour le cryptage de paramètres passés
 * en paramètres de l'URL
 */
public class SHA256 {
    /**
     * Hashage d'une chaine de caractère en SHA-256
     *
     * @param str
     *            Chaine de caractère à crypter
     * @return La chaîne cryptée
     */
    public static String getHashValue(String str) {
        MessageDigest md;
        StringBuffer sb = new StringBuffer();

        try {
            md = MessageDigest.getInstance("SHA-256");
            md.update(str.getBytes());

            byte byteData[] = md.digest();

            // Converti les byte en hexadécimal
            for (byte b : byteData) {
                sb.append(Integer.toString((b & 0xff) + 0x100, 16).substring(1));
            }
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }

        return sb.toString();
    }
}

```



```java
package toolbox;

import java.io.Serializable;

/**
 * Contient les méthodes pour la connexion et l'exécution de requêtes sur la
 * base de données.<br>
 * Les méthodes les plus importantes sont les suivantes :<br>
 * <br>
 * <ul>
 * <li><b>query</b> : Exécution d'une requête SQL de type SELECT, que l'on peut
 * récupérer dans un objet ResultSet.</li>
 * <li><b>exec</b> : Exécution d'une requête ne retournant aucun résultat, de
 * type INSERT, UPDATE, DELETE</li>
 * <li><b>Disconnect</b> : Destruction de la connexion active pour la libération
 * des ressources</li>
 * </ul>
 */
public class Sql implements Serializable {
    private static final long serialVersionUID = 1L;

    /** Contient une instance de la class SQL **/
    private static Sql instance = new Sql();
    /**
     * Conserve la dernière connexion effectuée.<br>
     * Pour la réinitialiser : Sql.Disconnect();
     **/
    private static Connection connection = null;
    /**
     * Chaine pointant vers le package nécessaire à l'utilisation des drivers
     * JDBC<br>
     * <li>mysql : com.mysql.jdbc.Driver (Connector/J requis)</li> <li>db2 :
     * com.ibm.db2.jcc.DB2Driver</li>
     */
    private final String JDBC_DRIVERS = "com.mysql.jdbc.Driver";
    /**
     * Nom du Système de gestion de base de données<br>
     * <br>
     * <li>Mysql</li><li>DB2</li><li>...</li>
     */
    private final String SGBD_NAME = "mysql";
    /**
     * Adresse pour atteindre le serveur de base de données<br>
     * <br>
     * <li><b>Production :</b> localhost</li><li><b>Développement :</b>
     * 10.76.210.172</li>
     */
    private final String SERVER = "localhost";
    /**
     * Port utilisé par la base de données<br>
     * DB2 : Trouver les ports utilisés : cmd > db2cmd > db2 list node
directory<br>
     * <br>
     * <li>DB2 PROD = :60000</li><li>MySQL : Pas besoin de spécifier (:3306)</li>
     */
    private final String PORT = "";
    /**
     * Nom de la base de données
     */
    private final String DB_NAME = "db_cccvstransfert";
    /**
     * Nom d'utilisateur pour la connexion à la base de données
```

```
73      */
74     private final String DB_USERNAME = "cccvstransfert";
75     /**
76      * Mot de passe de l'utilisateur DB_USERNAME
77      */
78     private final String DB_PASSWORD = "Kolat10s";
79     /**
80      * Contiendra le nombre de résultats retournés après l'exécution d'une
81      * requête SQL
82      */
83     private static int rowCount = 0;
84
85     /**
86      * CONSTRUCTEUR [Singleton]<br>
87      * On met le constructeur en privé par défaut car cet objet ne devrait pas
88      * être instancié plus d'une fois dans une méthode.
89      */
90     private Sql() {
91
92     }
93
94     /**
95      * Retourne une instance de la class {@link Sql}
96      *
97      * @return Sql L'instance
98      */
99     public static Sql getInstance() {
100        return instance;
101    }
102
103    /**
104     * Exécute la requête SQL passée en paramètre
105     *
106     * @param SQLQuery
107     *            Requête SQL à exécuter
108     * @return (Boolean) true si réussi
109     * @throws SQLException
110     */
111    public Boolean execSQLSimple(Connection connect, String SQLQuery)
112            throws SQLException {
113        Boolean state = false;
114        Statement stmt = null;
115        try {
116            // Exécution de la requête
117            stmt = connect.createStatement();
118            stmt.execute(SQLQuery);
119            state = true;
120        } catch (SQLException e) {
121            System.out.println(Utilities.getCurrentDate()
122                    + " - querySQL > La requête DB2 (select) a échoué -> "
123                    + SQLQuery);
124            System.out.println(e);
125            state = false;
126        }
127
128        return state;
129    }
130
131    /**
132     * Exécute la requête SQL et renvoie le résultat dans un SQLContainer
133     *
134     * @param query
```



```
135     *            Requête SQL à exécuter
136     * @return (SQLContainer) Un SQLContainer contenant le résultat de la
137     *         requête
138     * @throws SQLException
139     */
140    public SQLContainer execSQL(SimpleJDBCConnectionPool connectionContainer,
141            String SQLQuery) {
142        try {
143            // Exécution de la requête
144            FreeformQuery FFQ = new FreeformQuery(SQLQuery, connectionContainer);
145            if (FFQ.getCount() > 0) {
146                return new SQLContainer(FFQ);
147            } else {
148                return null;
149            }
150        } catch (SQLException e) {
151            System.out.println(Utilities.getCurrentDate()
152                    + " - querySQL > La requête DB2 (select) a échoué -> "
153                    + SQLQuery);
154            System.out.println(e);
155            return null;
156        }
157    }
158
159    // ----------------------------------------------------------------------------
    ---------------------------
160    /**
161     * Exécute une requête SQL de type SELECT et retourne les résultats dans un
162     * ResultSet.<br>
163     * Crée une connexion globale si aucune n'existe déjà
164     *
165     * @param query
166     *            Requête SQL (SELECT)
167     * @return (ResultSet) Résultat de la requête SELECT
168     * @author Dominique Roduit
169     */
170    public static ResultSet query(String query) {
171        Statement stmt = null;
172        ResultSet rs = null;
173
174        // Si la connexion n'existe pas encore on la crée, sinon on prend celle
175        // qui existe déjà
176        if (connection == null) {
177            connection = Sql.getInstance().Connect();
178        }
179
180        try {
181            stmt = connection.createStatement();
182            rs = stmt.executeQuery(query);
183
184            // Enregistrement du nombre de résultats retournés
185            rs.last();
186            rowCount = rs.getRow();
187            rs.beforeFirst();
188
189            System.out.println(Utilities.getCurrentDate() + " -- [OK] > "
190                    + query);
191        } catch (SQLException e) {
192            e.printStackTrace();
193            System.out.println(Utilities.getCurrentDate()
194                    + " -- [SQL query] La requête à échoué > " + query);
195        }
```

```
196
197         return rs;
198     }
199
200     /**
201      * Exécute une requête SQL de type UPDATE, INSERT, DELETE.<br>
202      * Crée une connexion globale si aucune n'est déjà existante.
203      *
204      * @param query
205      *            Requête SQL (UPDATE, INSERT, DELETE)
206      * @author Dominique Roduit
207      */
208     public static void exec(String query) {
209         Statement stmt = null;
210         String requestType = query.substring(0, query.indexOf(" "))
211                 .toUpperCase();
212
213         // Si la connexion n'existe pas encore on la crée, sinon on prend celle
214         // qui existe déjà
215         if (connection == null) {
216             connection = Sql.getInstance().Connect();
217         }
218
219         try {
220             stmt = connection.createStatement();
221             // Enregistrement du nombre de résultats retournés (0 si aucun
222             // résultat)
223             rowCount = stmt.executeUpdate(query);
224
225             System.out.println(Utilities.getCurrentDate() + " -- [OK] > "
226                     + query);
227         } catch (SQLException e) {
228             e.printStackTrace();
229             System.out.println(Utilities.getCurrentDate()
230                     + " -- [SQL exec] La requête à échoué > " + query);
231         }
232     }
233
234     /**
235      * Accesseur qui retourne le nombre de résultats retournés par les méthodes
236      * "query" et "exec".
237      *
238      * @return (int) Nombre de résultats de la dernière requête SQL exécutée
239      *         (tout types de requêtes).
240      */
241     public static int rowCount() {
242         return rowCount;
243     }
244
245     // ----------------------------------------------------------------------------
    ---------------------------
246     /**
247      * Créer la connexion à la base de donnée.<br>
248      * Pour des requêtes ne nécessitant pas de liaisons de données avec des
249      * composants VAADIN
250      *
251      * @return Connexion
252      * @author Dominique Roduit
253      */
254     public Connection Connect() {
255         Connection conn = null;
256         String connectionUri = "jdbc:" + SGBD_NAME + "://" + SERVER + PORT
```



```
257                 + "/" + DB_NAME;
258
259         // Création de l'instance de connexion
260         try {
261             System.out.println(Utilities.getCurrentDate()
262                     + " -- [Connexion réussie] " + SERVER + PORT + "/"
263                     + DB_NAME);
264             Class.forName(JDBC_DRIVERS).newInstance();
265             conn = DriverManager.getConnection(connectionUri, DB_USERNAME,
266                     DB_PASSWORD);
267             this.connection = conn;
268         } catch (Exception e) {
269             System.out.println(Utilities.getCurrentDate()
270                     + " -- [Connexion échouée] " + SERVER + PORT + "/"
271                     + DB_NAME);
272             e.printStackTrace();
273         }
274
275         return conn;
276     }
277
278     // ----------------------------------------------------------------------------
    ---------------------------
279     /**
280      * Créer la connexion à la base de donnée<br>
281      * Pour les requêtes qui retournent des données, tel que SELECT
282      *
283      * @return SimpleJDBCConnectionPool Pool simple de connexion JDBC
284      * @throws SQLException
285      */
286     public SimpleJDBCConnectionPool ConnectContainerTest() throws SQLException {
287         // Création de l'instance de connexion
288         String connectionUri = "jdbc:" + SGBD_NAME + "://" + SERVER + PORT
289                 + "/" + DB_NAME;
290         try {
291             System.out.println(Utilities.getCurrentDate()
292                     + " -- [Connexion réussie] " + SERVER + PORT + "/"
293                     + DB_NAME);
294             return new SimpleJDBCConnectionPool(JDBC_DRIVERS, connectionUri,
295                     DB_USERNAME, DB_PASSWORD);
296         } catch (Exception e2) {
297             System.out.println(Utilities.getCurrentDate()
298                     + " -- [Connexion échouée] " + SERVER + PORT + "/"
299                     + DB_NAME);
300             e2.printStackTrace();
301             return null;
302         }
303     }
304
305     /**
306      * Ferme la connexion active s'il en existe une
307      */
308     public static void Disconnect() {
309         try {
310             if (connection != null) {
311                 connection.close();
312                 connection = null;
313                 System.out.println(Utilities.getCurrentDate()
314                         + " -- [Déconnexion] Base de données");
315             }
316         } catch (SQLException e) {
317             System.out.println(e);
```

```
318         }
319     }
320 
321     /**
322      * Ferme la connexion passée en paramètre
323      *
324      * @param Connection
325      *            La connexion à détruire
326      */
327     public void Disconnect(Connection connect) {
328         try {
329             connect.close();
330         } catch (Exception e) {
331         }
332     }
333 
334     /**
335      * Ferme la connexion d'un container à la base de donnée
336      *
337      * @param connectionContainer
338      *            La connexion du container à détruire
339      */
340     public void DisconectContainer(SimpleJDBCConnectionPool connectionContainer) {
341         try {
342             connectionContainer.destroy();
343         } catch (Exception e1) {
344         }
345     }
346 
347 }
348 
```



```
1 package toolbox;
2 
3 import global.Global;
36 
37 /**
38  * Classe regroupant les fonctions importantes utilisées globalement dans toute
39  * l'application
40  *
41  * @author Roduit Dominique
42  *
43  */
44 public class Utilities {
45     /**
46      * Récupère la date et l'heure actuel
47      *
48      * @return Date Date actuelle au format 14:49:35.769
49      */
50     public static String getCurrentDate() {
51         String Date;
52         Date = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS ")
53                 .format(new Date());
54 
55         return Date;
56     }
57 
58     /**
59      * Formate une variable pouvant être nulle pour une requête sql
60      *
61      * @param in
62      *            Variable a formater (String)
63      * @return (String) Variable formatée
64      */
65     public static String formatSQL(String in) {
66         String Variable = in;
67         if (Variable == null) {
68             Variable = "NULL";
69         } else {
70             if (Variable.equals("")) {
71                 Variable = "NULL";
72             } else {
73                 String temp = "";
74                 temp = (Variable.indexOf("'") > -1) ? Variable.replaceAll("'",
75                         "''") : Variable;
76                 Variable = "'" + temp + "'";
77             }
78         }
79         return Variable;
80     }
81 
82     /**
83      * Formate une variable pouvant être nulle pour une requête sql
84      *
85      * @param in
86      *            Variable à formater (int)
87      * @return (String) Variable formatée
88      */
89     public static String formatSQL(Integer in) {
90         String Variable = null;
91         if (in == null) {
92             Variable = "NULL";
93         } else {
94             Variable = in.toString();
```

```
95          }
96          return Variable;
97      }
98
99      /**
100      * Formatte un boolean pour la base de données MySQL
101      *
102      * @param in
103      *            Le boolean
104      * @return Boolean formaté (0 ou 1)
105      */
106     public static String formatSQL(boolean in) {
107         return (in) ? "1" : "0";
108     }
109
110     /**
111      * Retourne le suffixe d'une adresse e-mail (ex. avs.vs.ch)
112      *
113      * @param email
114      *            Adresse e-mail
115      * @return Suffixe de l'adresse spécifiée
116      */
117     public static String getEmailSuffixe(String email) {
118         return email.substring(email.indexOf("@") + 1);
119     }
120
121     /**
122      * Indique si la personne est une interne ou non
123      *
124      * @param email
125      *            Adresse e-mail de la personne
126      * @return true si la personne est une interne
127      */
128     public static boolean isInternal(String email) {
129         String suffixe = getEmailSuffixe(email);
130         return suffixe.equals(Global.getParm("EMAIL_INTERNE_SUFFIXE"));
131     }
132
133     /**
134      * Retourne le nom d'une personne interne par rapport à son adresse e-mail
135      *
136      * @param email
137      *            Adresse e-mail de la personne
138      * @return Nom de la personne si c'est une interne
139      */
140     public static String getInternalName(String email) {
141         String name = "", prenom = "", nom = "";
142         if (isInternal(email)) {
143             name = email.substring(0, email.indexOf("@"));
144             name = name.replace(".", " ");
145
146             prenom = name.substring(0, 1).toUpperCase()
147                     + name.substring(1, name.indexOf(" ")).trim();
148             nom = name.replace(prenom.toLowerCase(), "").trim();
149             nom = nom.substring(0, 1).toUpperCase()
150                     + nom.substring(1, nom.length());
151             name = prenom + " " + nom;
152         }
153         return name;
154     }
155
156     /**
```



```
157      * Retourne le nom du système d'exploitation
158      *
159      * @return Nom du système d'exploitation
160      */
161     public static String getOS() {
162         if (Global.browser.isWindows()) {
163             return "Windows";
164         } else if (Global.browser.isMacOSX()) {
165             return "Mac OSX";
166         } else if (Global.browser.isLinux()) {
167             return "Linux";
168         } else {
169             return "Autre";
170         }
171     }
172
173     /**
174      * Retourne le nom du navigateur et sa version
175      *
176      * @return Nom du navigateur et sa version
177      */
178     public static String getBrowserAndVersion() {
179         if (Global.browser.isChrome()) {
180             return "Chrome " + Global.browser.getBrowserMajorVersion() + "."
181                     + Global.browser.getBrowserMinorVersion();
182         } else if (Global.browser.isOpera()) {
183             return "Opera " + Global.browser.getBrowserMajorVersion() + "."
184                     + Global.browser.getBrowserMinorVersion();
185         } else if (Global.browser.isFirefox()) {
186             return "Firefox " + Global.browser.getBrowserMajorVersion() + "."
187                     + Global.browser.getBrowserMinorVersion();
188         } else if (Global.browser.isSafari()) {
189             return "Safari " + Global.browser.getBrowserMajorVersion() + "."
190                     + Global.browser.getBrowserMinorVersion();
191         } else if (Global.browser.isIE()) {
192             return "Internet Explorer "
193                     + Global.browser.getBrowserMajorVersion();
194         } else {
195             return "Unknown";
196         }
197     }
198
199     /**
200      * Retourne l'extension d'un fichier en fonction de son nom
201      *
202      * @param name
203      *            Le nom du fichier
204      * @return Extension du fichier
205      */
206     public static String getExtension(String name) {
207         int dot = name.lastIndexOf(".");
208         return name.substring(dot + 1);
209     }
210
211     /**
212      * Formate la taille d'un fichier pour l'affichage
213      *
214      * @param size
215      *            Taille du fichier en octet
216      * @return Taille formatée
217      */
218     public static String formatSize(long size) {
```

```
219         String strSize = "";
220         long kb = 1024; // Kilo
221         long mb = 1024 * kb; // Mega
222         long gb = 1024 * mb; // Giga
223 
224         Locale currentLocale = Locale.getDefault();
225         NumberFormat formatter = NumberFormat.getNumberInstance(currentLocale);
226 
227         formatter.setMaximumFractionDigits(2);
228         // formatter.setMinimumFractionDigits(1);
229 
230         if (size < kb) {
231             strSize = size + " " + Global.i("CAPTION_OCTETS");
232         } else if (size < mb) {
233             strSize = formatter.format(size / kb) + " Ko";
234         } else if (size < gb) {
235             strSize = formatter.format(size / mb) + " Mo";
236         } else {
237             strSize = formatter.format(size / gb) + " Go";
238         }
239         return strSize;
240     }
241 
242     /**
243      * Retourne une taille en octet par rapport a une chaine de caractère
244      * formatée
245      *
246      * @param size
247      *            Taille formatée
248      * @return Taille en octet
249      */
250     public static long getFileSizeFromFormatedSize(String size) {
251         String strSize = size.substring(size.indexOf(" "), size.length())
252                 .trim();
253         String clean = size.replace(strSize, "").replace(".00", "")
254                 .replace(",00", "").trim();
255         long basic = Long.parseLong(clean);
256 
257         if (strSize.equals("Ko"))
258             basic = basic * 1024;
259         if (strSize.equals("Mo"))
260             basic = basic * 1024 * 1024;
261         if (strSize.equals("Go"))
262             basic = basic * 1024 * 1024 * 1024;
263 
264         return basic;
265     }
266 
267     /**
268      * Retourne le chemin vers l'image d'une extension de fichier
269      *
270      * @param ext
271      *            Extension du fichier
272      * @return Chemin et nom de l'image
273      */
274     public static Embedded getImgFromExtension(String ext) {
275         String exts =
    "ai,aiff,apk,asp,avi,bmp,c,class,cpp,css,doc,docx,exe,flac,flv,html,jar,java,jpg,js,
    "
276                 +
    "mkv,mov,mp3,mpeg,msi,odt,pdf,php,png,ppt,pptx,psd,rar,sh,sql,svg,tar,tiff,torrent,t
    xt,wmv,xls,xlsx,xml,zip,";
```



```
277 
278         String img = Global.PATH_THEME_RESSOURCES + "extensions/";
279         img += (exts.indexOf(ext.toLowerCase() + ",") < 0) ? "document" : ext
280                 .toLowerCase();
281         img += ".png";
282 
283         Embedded imgRes = new Embedded(null, new ThemeResource(img));
284 
285         return imgRes;
286     }
287 
288     /**
289      * Formate une date de la base de données MySQL
290      *
291      * @param date
292      *            Date à formater (format SQL : 2013-03-11 08:16:00.456)
293      * @param format
294      *            Format d'affichage de la date<br>
295      * <br>
296      *            <ul>
297      *            <li><b>dd</b> = jour</li>
298      *            <li><b>MM</b> = mois</li>
299      *            <li><b>MMMM</b> = mois en lettres</li>
300      *            <li><b>YY</b> = année</li>
301      *            <li><b>HH</b> = heure</li>
302      *            <li><b>mm</b> = minute</li>
303      *            <li><b>ss</b> = seconde</li>
304      *            </ul>
305      * @return La date formatée
306      */
307     public static String formatSQLDate(String date, String format) {
308         String year, month, day, hour, minute, second;
309         year = date.substring(0, 4);
310         month = date.substring(5, 7);
311         day = date.substring(8, 10);
312         hour = date.substring(11, 13);
313         minute = date.substring(14, 16);
314         second = date.substring(17, 19);
315 
316         ArrayList<String> monthList = new ArrayList<String>();
317         monthList.addAll(Arrays.asList(Global.i("CAPTION_JANVIER"),
318                 Global.i("CAPTION_FEVRIER"), Global.i("CAPTION_MARS"),
319                 Global.i("CAPTION_AVRIL"), Global.i("CAPTION_MAI"),
320                 Global.i("CAPTION_JUIN"), Global.i("CAPTION_JUILLET"),
321                 Global.i("CAPTION_AOUT"), Global.i("CAPTION_SEPTEMBRE"),
322                 Global.i("CAPTION_OCTOBRE"), Global.i("CAPTION_NOVEMBRE"),
323                 Global.i("CAPTION_DECEMBRE")));
324 
325         String dateF = format.replace("dd", day);
326         dateF = dateF.replace("MMMM", monthList
327                 .get(Integer.parseInt(month) - 1).toLowerCase());
328         dateF = dateF.replace("MM", month);
329         dateF = dateF.replace("YY", year);
330         dateF = dateF.replace("HH", hour);
331         dateF = dateF.replace("mm", minute);
332         dateF = dateF.replace("ss", second);
333 
334         return dateF;
335     }
336 
337     /**
338      * Formate une date de la base de données MySQL. Le format d'affichage est
```

```
339     * spécifié par défaut dans la table des paramètres
340     *
341     * @param date
342     *            Date au format MySQL
343     * @return Date Formatée
344     */
345    public static String formatSQLDate(String date) {
346        return formatSQLDate(date, Global.getParm("DATE_FORMAT"));
347    }
348
349    /**
350     * Formatte une date du format SQL vers le format donné et affiche en texte
351     * les dates Aujourd'hui et Demain.
352     *
353     * @param date
354     *            Date a formatter
355     * @param format
356     *            Format de la date
357     * @return Date formattée
358     */
359    public static String formatSQLDateWtText(String date, String format) {
360        if (formatSQLDate(date, "YY-MM-dd").equals(getDate("yyyy-MM-dd"))) {
361            return Global.i("CAPTION_TODAY") + " ("
362                    + zeroFill(Integer.parseInt(formatSQLDate(date, "HH")) + 1)
363                    + ":00)";
364        } else if (formatSQLDate(date, "YY-MM-dd").equals(
365                getDate("yyyy-MM-")
366                        + (zeroFill(Integer.parseInt(getDate("dd")) + 1)))) {
367            return Global.i("CAPTION_TOMORROW") + " ("
368                    + zeroFill(Integer.parseInt(formatSQLDate(date, "HH")) + 1)
369                    + ":00)";
370        } else {
371            return formatSQLDate(date, format);
372        }
373    }
374
375    /**
376     * Retourne le dossier d'upload du serveur. Si on est en local, la méthode
377     * retourne une chaine vide. <b>IMPORTANT !!!<b> - A n'utiliser que pour
378     * spécifier des URL !<br>
379     * <br>
380     * <b>Pour spécifier le chemin de destination pour l'upload d'un fichier,
381     * utiliser Global.UPLOAD_DIR</b>
382     *
383     * @return Dossier d'upload sur le serveur (ex. /files/)
384     */
385    public static String getUploadDir() {
386        String uploadDir = "";
387
388        uploadDir = Global.UPLOAD_DIR.substring(0,
389                Global.UPLOAD_DIR.length() - 1);
390        uploadDir = uploadDir.substring(uploadDir.lastIndexOf("/"),
391                uploadDir.length())
392                + "/";
393
394        return uploadDir;
395    }
396
397    /**
398     * Retourne l'URL d'un fichier
399     *
400     * @param file
```



```
401     *            Fichier pour lequel on veux obtenir l'URL
402     * @return URL du fichier
403     */
404    public static String getURLForFile(String file) {
405        String URL = Global.URL.toString();
406
407        if (Global.isDev) {
408            URL = URL.replace(":8443", "").replace(":8080", "")
409                    .replace("https:", "http:");
410        }
411
412        URL = URL.substring(0, URL.length() - 1);
413        URL = URL.substring(0, URL.lastIndexOf("/"));
414        URL += getUploadDir();
415
416        return URL + file;
417    }
418
419    /**
420     * Retourne la date actuelle au format désiré
421     *
422     * @param format
423     *            Format de date (ex. yyyyMMdd)
424     * @return Date au format désiré
425     */
426    public static String getDate(String format) {
427        DateFormat dateFormat = new SimpleDateFormat(format);
428        String date = dateFormat.format(new Date()).toString();
429        return date;
430    }
431
432    /**
433     * Supprime les accentuations d'une chaine de caractères
434     *
435     * @param arg
436     *            (String) Chaine dans laquelle supprimer les accents
437     * @return Chaine sans accents
438     */
439    public static String removeAccent(String s) {
440        String strTemp = Normalizer.normalize(s, Normalizer.Form.NFD);
441        Pattern pattern = Pattern.compile("\\p{InCombiningDiacriticalMarks}+");
442        return pattern.matcher(strTemp).replaceAll("");
443    }
444
445    /**
446     * Nettoye le nom d'un fichier<br>
447     * <br>
448     * <ul>
449     * <li>Supprime les accents</li>
450     * <li>Supprime les majuscules</li>Supprime les espaces</li>
451     * <li>Limite la taille de la chaine</li>
452     * </ul>
453     *
454     * @param name
455     *            Le nom du fichier à nettoyer
456     * @return Le nom du fichié proprement démuni de tout caractère embêtant
457     */
458    public static String cleanFileName(String name) {
459        String filename = removeAccent(name).replace(" ", "_").toLowerCase();
460
461        filename = filename.replaceAll("[^a-zA-Z0-9]", "");
462
```

```
463         if (filename.length() > 130)
464             return filename.substring(130);
465         else
466             return filename;
467     }
468 
469     /**
470      * Suppression du code HTML dans une chaine
471      *
472      * @param htmlString
473      *            Chaine de caractère contenant le code HTML
474      * @return Chaine parsée
475      */
476     public static String htmlentities(String htmlString) {
477         return htmlString.replace("<", "").replace(">", "");
478     }
479 
480     /**
481      * Formate une unité avec un 0 devant
482      *
483      * @param number
484      *            Nombre à formatter
485      * @return Nombre formatté
486      */
487     public static String zeroFill(int number) {
488         return (number < 10) ? "0" + number : Integer.toString(number);
489     }
490 
491     /**
492      * Retourne le nom de l'archive d'un dossier par rapport à son nom
493      *
494      * @param folder_name
495      *            Nom du dossier
496      * @return Nom de l'archive
497      */
498     public static String getFolderArchiveName(String folder_name) {
499         return Global.getParm("PREFIXE_ARCHIVE_FOLDER")
500                 + Utilities.cleanFileName(folder_name) + "_"
501                 + Utilities.getDate("yyyyMMdd") + ".zip";
502     }
503 
504     /**
505      * Retourne le nom de l'archive d'un dossier par rapport à son nom et sa
506      * date de création
507      *
508      * @param folder_name
509      *            Nom du dossier
510      * @param folder_creation_date
511      *            Date de création du dossier
512      * @return Nom de l'archive
513      */
514     public static String getFolderArchiveName(String folder_name,
515             String folder_creation_date) {
516         return Global.getParm("PREFIXE_ARCHIVE_FOLDER")
517                 + Utilities.cleanFileName(folder_name) + "_"
518                 + Utilities.formatSQLDate(folder_creation_date, "YYMMdd")
519                 + ".zip";
520     }
521 
522     /**
523      * Remplace tous les paramètres d'une chaine de la forme %PARAMETRE% par
524      * leur valeurs respectives
```



```
525      *
526      * @param content
527      *            Chaine de caractère dans laquelle se trouve les paramètres
528      * @return Chaine avec les valeurs des paramètres
529      */
530     public static String parmConverter(String content) {
531         Pattern p = Pattern.compile("%(.*)%");
532         Matcher m = p.matcher(content);
533 
534         String[] parms = null;
535         while (m.find())
536             parms = m.group().split(" ");
537 
538         if (parms != null) {
539             for (int i = 0; i < parms.length; i++) {
540                 content = content.replace(parms[i],
541                         Global.getParm(parms[i].replace("%", "")));
542             }
543         }
544         return content;
545     }
546 
547     /**
548      * Renvoi le nom de fichier formatté comme nous le voulons pour
549      * l'enregistrement sur le disque.
550      *
551      * @param originalFileName
552      *            Nom du fichier original
553      * @return Nom du fichier tel qu'il sera enregistré sur le disque
554      */
555     public static String getNewFileName(String originalFileName) {
556         String date = Utilities.getDate("yyyyMMddHHmmss");
557         String newName = Utilities.cleanFileName(originalFileName.replace(
558                 Utilities.getExtension(originalFileName), ""))
559                 + "_"
560                 + date
561                 + "." + Utilities.getExtension(originalFileName);
562         return newName;
563     }
564 }
565 
```

```java
package toolbox;

import java.io.BufferedOutputStream;

/**
 * Création d'un fichier d'archive au format ZIP, compression et ajours de
 * fichiers à l'archive.
 *
 * @author Fobec 2011
 */
public class ZipFileWritter {
    /** Flux de l'archive zip **/
    private ZipOutputStream zos;

    /**
     * Constructeurr : creation d'une nouvelle archive
     *
     * @param zipFile
     *            Nom du fichier ZIP à créer
     * @throws FileNotFoundException
     */
    public ZipFileWritter(String zipFile) throws FileNotFoundException {
        FileOutputStream fos = new FileOutputStream(zipFile);
        // ajout du checksum
        CheckedOutputStream checksum = new CheckedOutputStream(fos,
                new Adler32());
        this.zos = new ZipOutputStream(new BufferedOutputStream(checksum));
    }

    /**
     * Ajoute un fichier au fichier zip
     *
     * @param fileName
     *            Chemin vers le fichier
     * @throws FileNotFoundException
     * @throws IOException
     */
    public void addFile(String fileName) throws FileNotFoundException,
            IOException {
        FileInputStream fis = new FileInputStream(fileName);
        int size = 0;
        byte[] buffer = new byte[1024];

        // Ajouter une entree à l'archive zip
        File file = new File(fileName);
        ZipEntry zipEntry = new ZipEntry(file.getName());
        this.zos.putNextEntry(zipEntry);

        // copier et compresser les données
        while ((size = fis.read(buffer, 0, buffer.length)) > 0) {
            this.zos.write(buffer, 0, size);
        }

        this.zos.closeEntry();
        fis.close();
    }

    /**
     * Fermer le fichier flux de création du fichier zip
     *
     * @throws IOException
     */
```



```java
    public void close() throws IOException {
        this.zos.close();
    }
}
```

CSS

STYLES.CSS

Styles des layouts, composants personnalisés, modification de l'apparence des composants natifs de Vaadin. Styles propres à internet explorer inclus.

EASYUPLOADS.CSS

Apparence personnalisée du module de transfert et de la zone de drag&drop.

```css
@import "../runo/styles.css";
/*---------------------------------------
---------------- CLEARFIX
---------------------------------------*/
html,body {
    width: 100%;
    height: 100%;
    margin:0;
}
/* --- Connexion ---*/
.connexion .v-body {
    background: #eee;
}
.v-login-form {
    color: white;
}
.v-logo {
    background: url('../img/logoCCCVs.png') no-repeat;
}
.containerLogo {
    margin-top: 8px;
}
.velConnexion {
    margin-top: 8px;
}
.connexion .v-window {
    background: transparent;
}
.connexion .v-window-wrap {
    border: 1px solid #808386;
}
.black .v-button-caption {
    color: #C9CCCE;
    text-shadow: 0 -1px 0 #121314;
    font-size: 11px;
    font-weight: bold;
    line-height: 16px;
}

/*---------------------------------------
---------------- HEADER
---------------------------------------*/
.v-header {
    background: url(../img/header_bg.jpg) repeat-x #151412;
}
.v-header-logo {
}
.v-header-menu {
    text-align: right;
    color: #c0c0be;
}
.v-header-menu .v-button-wrap { /* .v-header-menu .v-button-caption */
    background: #000;
    border: none;
    color: #c0c0be;
    text-shadow: none;
    border-radius: 6px;
}
.v-header-menu .v-button-caption {
    background: none;
}
.v-header-menu .v-button-wrap:hover {
```



```css
    background : #4c4c4b;
    color: white;
}
.v-header-menu .v-button { /* , .v-disabled.v-button */
    background-image: none;
}
/*---------------------------------------
---------------- MENU
---------------------------------------*/
.v-menu-content {
    background: #f2f2f3;
}
.v-menu-ribbon {
    background: url(../img/menu-ribbon-bg.jpg) repeat-x #d11e00;
}
.v-menu-ribbon-lbl {
    color: #f2dbda;
    padding-left: 16px;
    font-size: 14px;
    font-family: Verdana, Arial, sans-serif;
    background: url(../img/download_logo.png) no-repeat 210px 50%;
    line-height: 47px;
    text-shadow: 0 1px 0 #A90101;
    font-weight: bold;
}
.v-menu-ribbon.expand .v-menu-ribbon-lbl {
    background: url(../img/download_logo.png) no-repeat 255px 50%;
}
.v-accordion-icon .v-caption {
    background-position: 240px 50%;
    padding-left:10px;
}
.v-accordion-icon .v-accordion-item-open .v-accordion-item-caption .v-caption {
    background-position: 240px 50%;
}
.v-accordion-icon .v-caption .v-icon {
    margin-top: -3px;
    padding-right: 8px;
}
.v-accordion-icon .v-caption .v-captiontext {
    font-size: 14px;
}
.v-menu-item-folder, .v-menu-item-contact {
    padding-left: 26px;
    margin-left: 8px;
}
.v-menu-item-folder {
    background: url(../img/directory.png) no-repeat 0px 2px;
}
.v-label-archive {
    background: url(../img/directory_archive.png) no-repeat 0px 2px;
}
.v-menu-item-contact {
    background: url(../img/home.png) no-repeat 4px 2px;
    padding-left: 30px;
    margin-left: 4px;
}
.contact-extern {
    background: url("../img/icon-contact.png") no-repeat 0px 0px;
}
.no-resizable .v-table-resizer {
    width: 0;
```

```
}

.admin-menu .v-button .v-icon {
    float:left;
    margin-right: 0;
}
/*---------------------------------------
---------------- FOOTER
---------------------------------------*/
.v-footer {
    background: #fff;
    border-top: 5px solid #b8bcbf;
}
.v-footer-toogle {
    border-bottom: 1px solid #b8bcbf;
}
.v-caption-v-footer-toogle-text {
    padding-left: 10px;
}
.v-caption-v-footer-toogle-text .v-icon {
    padding-top: 2px;
}
.v-caption-v-footer-toogle-text .v-captiontext {
    padding-left: 4px;
}
.v-footer-toogle-button .v-button-wrap {
    border: none;
    background: none;
}
.v-caption-footer-language {
    margin-right: 5px;
}
/*---------------------------------------
---------------- BODY
---------------------------------------*/
.v-body {
    background: #fff;
}
.v-body-menu {
    background: #f2f2f3;
    border-right: 1px solid #d5d2d0;
}
.v-body-ribbon {
    border-bottom: 1px solid #b8bcbf;
    padding-left: 18px;
}
.v-body-ribbon-menu {
}

.ribbon-right-alignment {
    position:absolute;
    right: 16px;
    top: 8px;
}
/* Hack IE */
.v-ie .ribbon-right-alignment {
    position:relative;
    top:0px;
    right:0px;
}
.ribbon-right-alignment .v-button-wrap {
    padding-left: 12px;
```



```
}
.ribbon-right-alignment  .v-icon {
    margin-right: 5px;
}

.finger {
    background: url(../img/finger.png) no-repeat;
    margin-left: 10px;
}
/*---------------------------------------
---------------- STYLES GENERAUX
---------------------------------------*/
.display-none {
    display: none;
}
.display {
    display: block;
}
.inline {
    display: inline;
}
.v-caption-inline, .v-caption-inline div {
    display: inline;
}
.v-textfield-inline {
    margin-bottom: 6px;
}
.align-center {
    text-align:center;
}
.cursor-pointer {
    cursor: pointer;
}
.v-caption-schema-illustration {
    margin-top: 35px;
    margin-bottom: 40px;
    font-size: 15px;
    font-weight: bold;
    color: #ba5904;
    font-family: Verdana, Arial, sans-serif;
}

.content-area-about {
    width: 750px;
    margin:auto;
    margin-top: 8px;
    padding: 10px;
    border:1px solid #ddd;
    border-top: 4px solid #ddd;
    border-bottom: 4px solid #ddd;
    border-radius: 10px;
}
.caption-generation {
    font-weight: bold;
    padding: 3px 12px;
    margin-top: 8px;
    background: #333;
    border-radius: 5px;
    color: #eee;
}
.caption-uploadsize {
    position:absolute;
```

```
    left:20px;
    background: #fff;
    padding: 2px 8px;
    border-radius: 6px;
    border:1px solid #ccc;
    color:#666;
    margin-top: -12px;
}
/*--- Dossiers ---*/
.info-folder {
    background: white;
}
.info-folder .infos {
    padding: 0px;
}
.info-folder h2 {
    margin: 0;
    color: #000;
    background: #eee;
    letter-spacing: 1px;
    padding: 3px 0px;
    font-size: 17px;
    text-align:center;
}
.info-folder .dates {
    font-size: 11px;
    color: #777;
    line-height: 1.5em;
}
.info-folder .dates table {
    text-align:center;
    width:100%;
}
.info-folder .filesize {
    font-weight: bold;
    margin-bottom: 4px;
    border-bottom: 1px solid #ddd;
    padding: 3px 0px;
    text-align:center;
    background: #eee;
}
.info-folder .title {
    font-weight: bold;
    padding: 2px 8px 3px;
    padding-bottom: 3px;
    background: #eee;
    color:black;
    margin: 6px 0;
}
.info-folder ul {
    list-style: square;
    color: black;
    padding-left:33px;
}
.info-folder .folder-status {
    background: url(../img/done.png) no-repeat 10px 50% #EEE;
    border-top: 1px solid #DDD;
    color: #333;
    padding: 4px 0px;
    height: 34px;
    color: #070;
    text-align: center;
```



```
    font-size: 15px;
    font-weight: bold;
    line-height: 2.3em;
}
.info-folder .archive {
    background: url(../img/warning.png) no-repeat 10px 50% #EEE;
    color: #800;
}
.info-folder .description {
    font-size: 11px;
    color: #444;
    padding: 5px 10px;
    padding-bottom: 1px;
    margin-top: 4px;
    border-top: 1px solid #ddd;
}
.recipients li img {
    margin-top:1px;
    margin-left:4px;
    position:absolute;
}
.recipients li.action {
    transition: all 0.25s linear;
}
.recipients li.action:hover {
    text-indent: 10px;
    font-weight: bold;
}
.vl-recipients {
    padding: 1px 8px;
}
.vl-recipients .v-button {
    width: 285px;
}
.vl-recipients .v-button .v-icon {
    position:absolute;
    left: 12px;
}
.vl-recipients .bt-active .v-button-wrap:hover {
    background: url("../img/btSmallHover.png") repeat-x;
    color: #222;
}
.vl-recipients .bt-non-active {
    color: #666;
    cursor: default;
}
.vl-recipients .bt-non-active .v-button-wrap {
    border: 1px solid #ccc;
}
.winStat .v-verticallayout {
    overflow: auto;
}
.winStat .title {
    margin-bottom: 6px;
    font-weight: bold;
    font-size: 18pt;
}
.get-link-textarea {
    text-align: center;
    line-height: 40px;

}
```

```
/*--------------------------------------
------- COMPOSANTS NATIFS [VAADIN]
--------------------------------------*/
/*--- #Tree ---*/
.v-tree {
    color: #15191b;
}
.v-tree-node {
    background: url("../img/collapsed.png") no-repeat scroll 2px 1px transparent;
}
.v-tree-node-expanded {
    background: url("../img/expanded.png") no-repeat scroll 2px 1px transparent;
}
.v-tree-node-caption span {
    padding: 0 4px;
}
/*--- #Table ---*/
.v-table-header-wrap, .v-table-header, .v-table-header-cell-asc
  .v-table-sort-indicator, .v-table-header-cell-desc .v-table-sort-indicator {
    height: 25px;
}
.v-table-caption-container {
    padding: 4px 2px 9px 0;
    font-size: 13px;
}
.v-table-column-selector {
    margin: -23px 0 0;
}
/*--- #TextField ---*/
.v-textfield {
    border: 2px solid #b8bcbf;
    border-radius: 5px;
    height: 18px;
}
/*--- #Contextmenu ---*/
.v-contextmenu {
    border-radius: 5px;
    background: #fdfdfd;
    padding: 3px 0;

    color: #383b3d;
    font-family:'Open Sans', Arial, Helvetica, sans-serif;
    font-size: 13px;
}
.v-contextmenu .gwt-MenuBar {
    border:none;
}
.v-shadow {
    opacity: 0.6;
}
.v-contextmenu .gwt-MenuItem-selected div {
    background: #d8290a;
}
.v-contextmenu .gwt-MenuItem div {
    padding: 4px 20px 4px 8px;
}
.v-contextmenu .gwt-MenuItem {
    padding: 0;
}
.v-form-errormessage {
    border: 1px solid #EE5500;
    border-radius: 5px;
```



```
    padding: 2px 0px;
    background-color: #fee;
    background-position: 5px 4px;
    margin-top:0;
    margin-bottom: 10px;
    text-align:center;
}

.v-multifileupload-dropzone-hide {
    display: none;
}
.v-label-slct-files {
    display: block;
    background: white;
    border: 2px solid #ccc;
    color: #bbb;
    font-size: 22px;
    text-align:center;
    border-radius: 6px;
    width: 100%;
    padding-top: 10px;
}

.v-progressindicator-wrapper {
    border-radius: 4px;
}
.v-progressindicator-success .v-progressindicator-indicator {
    background: url(../img/piSuccess.png) repeat-x transparent;
}
.v-progressindicator-super {
    width: 240px;
    position:absolute;
    left: 135px;
    margin-top: -4px;
}
.v-progressindicator-super .v-progressindicator-wrapper {
    height: 9px;
}
.v-progressindicator-super .v-progressindicator-indicator {
    background: url(../img/piSuper.png) repeat-x transparent;
    height: 9px;
}
.v-Notification-tray h1 {
    padding: 3px 10px 4px;
    border-radius: 3px;
    font-size: 14px;
}
.v-Notification-tray p {
    font-size: 13px;
}
.v-formlayout-captioncell {
    vertical-align: top;
}
.v-textarea {
    border: 2px solid #b8bcbf;
    border-radius: 5px;
}
.table-editable .v-textfield {
    border: 1px solid #bbb;
    margin: 1px;
}
.gwt-MenuItem .v-disabled {
```

```
    display: none
}

.v-table-body-noselection .v-table-row.v-selected:hover, .v-table-body-noselection
  .v-table-row-odd.v-selected:hover {
    background-color: #57A7ED;
}
div.v-window-header {
    color: #333;
}
.v-window-header .v-icon {
    margin-right: 8px;
    margin-top: -4px;
}
input.v-textfield-readonly, textarea.v-textarea-readonly {
    border: 2px solid #B8BCBF;
    border-radius: 5px;
    background: url("../runo/textfield/img/bg.png") repeat-x scroll 0 0 #FFFFFF;
}
```



```
.v-multifileupload-dropzone {
    background: url(../../themes/img/drophere.png) no-repeat 50% 50% #fff;
    border: 3px dashed #ccc;
    text-align:center;
    display:block;
    vertical-align: middle;
    font-size: 25px;
    color: #aaa;
    line-height: 8.1em;
    margin-top: 10px;
    margin-left: 3px;
}
.v-multifileupload-uploads {
    display:block;
    vertical-align: middle;
}
.v-multifileupload-uploads .v-upload {
    cursor: pointer;
}
.v-upload .swfupload {
    position: absolute;
    z-index: 1;
}
.v-ddwrapper-over {
    background: url(../../themes/img/drophereHover.png) no-repeat 50% 50% #eff;
    color: black;
    border: 3px dashed #acc
}
.v-ddwrapper-over, .v-multifileupload-dropzone {
    width: 660px;
    min-width:660px;
    max-width:660px;

    height:195px;
    min-height: 195px;
    max-height: 195px;
}

```

SQL

DB_CCCVSTRANSFERT.SQL

Création de la base de données et des différentes tables.

Ce script ne contient pas les autres requêtes telles que la création des contraintes et insertions car ces parties ont simplement été générées par le logiciel MySQL WorkBench.

```sql
-- Suppression de la base de données si elle existe
DROP SCHEMA IF EXISTS db_cccvstransfert;
CREATE SCHEMA IF NOT EXISTS db_cccvstransfert;

-- Utilisation de la base de données
USE db_cccvstransfert;

-- -----------------------------------------------------

--
-- Création de la table trans_app_param
--

CREATE TABLE IF NOT EXISTS trans_app_param (
  PKNoParam int(11) NOT NULL AUTO_INCREMENT,
  parm_key varchar(45) DEFAULT NULL COMMENT 'Clé/Nom du paramètre',
  parm_value text COMMENT 'Valeur du paramètre',
  parm_description text COMMENT 'Description du paramètre',
  PRIMARY KEY (PKNoParam)
) ENGINE=InnoDB  COMMENT='Paramètres de l''application';

--
-- Création de la table trans_contacts
--

CREATE TABLE IF NOT EXISTS trans_contacts (
  PKNoContact int(11) NOT NULL AUTO_INCREMENT,
  FKNoUser int(11) NOT NULL COMMENT 'Assignation du contact à un utilisateur de
  l''application',
  contact_name varchar(25) DEFAULT NULL COMMENT 'Nom ou description du contact',
  contact_mail varchar(255) DEFAULT NULL COMMENT 'Adresse e-mail du contact',
  contact_creation_date datetime DEFAULT NULL COMMENT 'Date de création du contact',
  contact_internal tinyint(1) DEFAULT NULL COMMENT 'true : Contact interne à la CCCVs /
  false : Contact externe',
  PRIMARY KEY (PKNoContact),
  KEY ContactOfUser_idx (FKNoUser)
) ENGINE=InnoDB
COMMENT='Contacts des utilisateurs';

--
-- Structure de la table trans_files
--

CREATE TABLE IF NOT EXISTS trans_files (
  PKNoFile int(11) NOT NULL AUTO_INCREMENT,
  FKNoFolder int(11) DEFAULT NULL COMMENT 'Assignation du fichier à un dossier de transfert',
  file_name varchar(150) DEFAULT NULL COMMENT 'Nom du fichier (sans le chemin)',
  file_rename varchar(60) DEFAULT NULL COMMENT 'Nom du fichier spécifié par l''utilisateur',
  file_size bigint(20) DEFAULT NULL COMMENT 'Taille du fichier [en Octets]',
  file_extension varchar(12) DEFAULT NULL COMMENT 'Extension du fichier',
  file_description text COMMENT 'Description du fichier spécifiée par l''utilisateur',
  PRIMARY KEY (PKNoFile),
  KEY FileFromFolder_idx (FKNoFolder)
) ENGINE=InnoDB
COMMENT='Fichiers d''un transfert';
```



```sql
--
-- Création de la table trans_folders
--

CREATE TABLE IF NOT EXISTS trans_folders (
  PKNoFolder int(11) NOT NULL AUTO_INCREMENT,
  FKNoUser int(11) DEFAULT NULL COMMENT 'Assignation du dossier à un utilisateur de
  l''application',
  folder_name varchar(50) DEFAULT NULL COMMENT 'Nom du dossier de transfert',
  folder_creation_date datetime DEFAULT NULL COMMENT 'Date de création du dossier',
  folder_expiration datetime DEFAULT NULL COMMENT 'Date de destruction du dossier',
  folder_archive tinyint(1) DEFAULT NULL COMMENT 'true : le fichier est archivé, il a été
  détruit automatiquement / false : Le dossier est toujours valable',
  PRIMARY KEY (PKNoFolder),
  KEY FolderOfUser_idx (FKNoUser)
) ENGINE=InnoDB
COMMENT='Conteneur des fichiers d''un transfert';

--
-- Création de la table trans_journal_connections
--

CREATE TABLE IF NOT EXISTS trans_journal_connections (
  PKNoJourConnection int(11) NOT NULL AUTO_INCREMENT,
  joco_mail varchar(255) DEFAULT NULL COMMENT 'Adresse e-mail de la personne qui se connecte',
  joco_date datetime DEFAULT NULL COMMENT 'Date d''insertion de l''entrée',
  joco_ip varchar(15) DEFAULT NULL COMMENT 'Adresse IP de la personne qui se connecte',
  joco_os varchar(45) DEFAULT NULL COMMENT 'Système d''exploitation de la personne qui se
  connecte',
  joco_browser varchar(45) DEFAULT NULL COMMENT 'Navigateur de la personne qui se connecte',
  joco_action varchar(100) DEFAULT NULL COMMENT 'Par ex. erreur générée (Connexion réussie,
  échec, ...)',
  joco_comment text COMMENT 'Détail sur l''action (Par ex. : N à tenté de se connecter mais
  la connexion a échouée)',
  PRIMARY KEY (PKNoJourConnection)
) ENGINE=InnoDB
COMMENT='JOURNALISATION : Connexions utilisateurs';

--
-- Création de la table trans_journal_download
--

CREATE TABLE IF NOT EXISTS trans_journal_folders (
  PKNoJourFolder int(11) NOT NULL AUTO_INCREMENT,
  FKNoContact int(11) DEFAULT NULL COMMENT 'Liaison vers le contact qui a téléchargé le
  fichier (et non l''auteur du dossier !)',
  FKNoFolder int(11) DEFAULT NULL COMMENT 'Liaison vers le dossier contenant le fichier
  téléchargé',
  FKNoFile int(11) NOT NULL COMMENT 'Liaison vers le fichier téléchargé',
  joufo_action varchar(60) DEFAULT NULL,
  joufo_date datetime DEFAULT NULL COMMENT 'Date du téléchargement',
  PRIMARY KEY (PKNoJourDownload),
  KEY DownloadedByContact_idx (FKNoContact),
  KEY ContentFromFolder_idx (FKNoFolder),
  KEY DownFileFromFolder_idx (FKNoFile)
) ENGINE=InnoDB
COMMENT='JOURNALISATION : Téléchargements des invités aux dossiers';
```

```
--
-- Création de la table trans_recipients
--

CREATE TABLE IF NOT EXISTS trans_recipients (
  PKNoRecipient int(11) NOT NULL AUTO_INCREMENT,
  FKNoFolder int(11) DEFAULT NULL COMMENT 'Lien vers le dossier de transfert',
  FKNoContact int(11) DEFAULT NULL COMMENT 'Lien vers le destinataire du transfert',
  PRIMARY KEY (PKNoRecipient),
  KEY RecipientForFolder_idx (FKNoFolder),
  KEY ContactLinkRecipient_idx (FKNoContact)
) ENGINE=InnoDB
COMMENT='Destinataires d''un transfert';

--
-- Création de la table trans_translate
--

CREATE TABLE IF NOT EXISTS trans_translate (
  PKNoTranslate int(11) NOT NULL AUTO_INCREMENT,
  trans_label varchar(45) DEFAULT NULL COMMENT 'Mot clé pour un mot',
  trans_fr text COMMENT 'Texte en français',
  trans_de text COMMENT 'Texte en allemand',
  PRIMARY KEY (PKNoTranslate)
) ENGINE=InnoDB
COMMENT='Traductions de l''application';

--
-- Création de la table trans_users
--

CREATE TABLE IF NOT EXISTS trans_users (
  PKNoUser int(11) NOT NULL AUTO_INCREMENT,
  user_mail varchar(255) DEFAULT NULL COMMENT 'Adresse e-mail de l''utilisateur',
  user_pass varchar(260) DEFAULT NULL COMMENT 'Mot de passe de l''utilisateur (cryptage :
  SHA256)',
  user_internal tinyint(1) DEFAULT NULL COMMENT 'true : l''utilisateur travaille à la CCCVs
  / false : L''utilisateur viens de l''extérieur',
  user_validation tinyint(1) DEFAULT NULL COMMENT 'true : L''adresse e-mail est validée /
  false : La validation est en attente, l''utilisateur ne peut pas se connecter',
  user_validation_date datetime DEFAULT NULL COMMENT 'Date à laquelle l''utilisateur a
  validé son adresse e-mail',
  user_langue varchar(2) DEFAULT NULL COMMENT 'Langue de l''utilisateur',
  user_admin tinyint(1) NOT NULL DEFAULT '0' COMMENT 'Indique si l''utilisateur est un
  administrateur ou pas',
  PRIMARY KEY (PKNoUser)
) ENGINE=InnoDB
COMMENT='Utilisateurs de l''application';
```

REMOVEARCHIVES

Projet JAR séparé de l'interface utilisateur pour la suppression automatique des fichiers et l'archivage des dossiers.

Ce projet utilise plusieurs copies des classes de l'interface utilisateur, par conséquent, je n'ai inclus que les classes spécifiques à ce projet qui n'ont pas déjà été présentées précédemment.

```java
package main;

import java.io.File;

/**
 * Projet de suppression des fichiers et d'archivage des dossiers expirés de
 * l'application CCCVs-Transfert.<br>
 * Cette classe contient la méthode exécutable qui est lancée lors du lancement
 * de l'archive JAR.<br>
 * Vous trouvez ce projet sur le serveur de production au format JAR, sur
 * /opt/tomcat7/work/CCCVsTransfert-cron/.<br>
 * Lorsqu'une nouvelle version doit être exportée, il faut choisir Jar
 * exécutable !
 *
 * @author Dominique Roduit
 *
 */
public class RemoveArchive {
    /**
     * Méthode appelée lors de l'exécution du projet
     */
    public static void main(String[] args) {
        // Date d'expiration arrive bientôt -> Notification pour avertissement
        RemoveArchive.warningBeforeExpiration();
        // Date d'expiration arrivée, suppression
        RemoveArchive.deleteFilesFolderAndMail();
    }

    /**
     * Avertissement des destinataires et de l'auteur du transfert.<br>
     * On leur envoie un mail qui leur dit que leur dossier va bientôt expirer.
     */
    public static void warningBeforeExpiration() {
        // Sélection des dossiers qui vont être supprimés dans moins de 24h et
        // qui ont une durée de vie > 1 jour
        ResultSet folder = Sql.query("SELECT * FROM trans_folders "
                + "LEFT JOIN trans_users ON PKNoUser=FKNoUser "
                + "WHERE folder_expiration<ADDDATE(now()," 
                + Global.getParm("DAY_BEFORE_FOLDER_DELETION") + ") " + // Qui
                                                                        // sera
                                                                        // supprimé
                                                                        // dans
                                                                        // < 24h
                "AND DATEDIFF(folder_expiration,folder_creation_date)>1 " + // Qui
                                                                           // a
                                                                           // une
                                                                           //
                                                                           durée
                                                                           // de
                                                                           // vie
                                                                           // >
                                                                           // 1
                                                                           // jour
                "AND DATEDIFF(folder_expiration,now())>0 " + // Qui n'expire pas
                                                             // aujourd'hui
                                                             // même
                "AND folder_archive=0 " + // Qui ne sont pas encore archivés
                "AND HOUR(folder_expiration)=HOUR(now()) " // Qui vont expirés
                                                           // pile un jour
                                                           // après maintenant
                                                           // (A l'heure
                                                           // d'exécution du
                                                           // script)
        );

        try {
            if (folder.first()) {
                folder.beforeFirst();

                while (folder.next()) {
                    Folder modelFolder = new Folder(
                            folder.getInt("PKNoFolder"),
                            folder.getInt("FKNoUser"),
                            folder.getString("folder_name"),
                            folder.getString("folder_description"),
                            folder.getString("folder_creation_date"),
                            folder.getString("folder_expiration"),
                            folder.getBoolean("folder_archive"),
                            folder.getString("user_mail"));

                    // Envoi d'un mail à l'AUTEUR
                    Mailer.sendMailAuthorBeforeRemovingFolder(
                            folder.getString("user_mail"), modelFolder,
                            folder.getString("user_langue"));

                    // Envoi d'un mail aux CONTACTS AYANT ACCES AU DOSSIER
                    ResultSet contact = Sql
                            .query("SELECT * FROM trans_recipients "
                                    + "LEFT JOIN trans_contacts ON
PKNoContact=FKNoContact "
                                    + "WHERE FKNoFolder="
                                    + folder.getString("PKNoFolder"));
                    while (contact.next()) {
                        Contact modelContact = new Contact(
                                contact.getInt("PKNoContact"),
                                contact.getString("contact_name"),
                                contact.getString("contact_mail"),
                                contact.getBoolean("contact_internal"));

                        Mailer.sendMailRecipientBeforeRemovingFolder(
                                modelContact, folder.getString("user_mail"),
                                modelFolder);
                    }
                }

            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    /**
     * Suppression des fichiers et archivage des dossiers expirés.<br>
     * Envoi des mails aux destinataires et à l'auteur pour les prévenir de
     * l'expiration.
     */
    public static void deleteFilesFolderAndMail() {
        // Sélection des dossiers dont la date d'expiration est inférieur à la
        // date actuelle
        ResultSet folder = Sql.query("SELECT * FROM trans_folders "
                + "LEFT JOIN trans_users ON PKNoUser=FKNoUser "
                + "WHERE folder_expiration<=now() AND folder_archive=0");

        int nbDossier = Sql.rowCount();
```

```
132
133         try {
134             String sqlFolder = "";
135
136             if (folder.first()) {
137                 folder.beforeFirst();
138
139                 while (folder.next()) {
140                     Folder modelFolder = new Folder(
141                             folder.getInt("PKNoFolder"),
142                             folder.getInt("FKNoUser"),
143                             folder.getString("folder_name"),
144                             folder.getString("folder_description"),
145                             folder.getString("folder_creation_date"),
146                             folder.getString("folder_expiration"),
147                             folder.getBoolean("folder_archive"),
148                             folder.getString("user_mail"));
149
150                     // Archivage du dossier
151                     System.out.println("\n");
152                     sqlFolder = "UPDATE trans_folders SET folder_archive=1 WHERE
    PKNoFolder="
153                             + folder.getString("PKNoFolder");
154                     Sql.exec(sqlFolder);
155
156                     // Suppression des fichiers
157                     ResultSet file = Sql
158                             .query("SELECT * FROM trans_files WHERE FKNoFolder="
159                                     + folder.getString("PKNoFolder"));
160                     while (file.next()) {
161                         File f = new File(Global.UPLOAD_DIR
162                                 + file.getString("file_name"));
163                         modelFolder.addFile_list(file.getString("file_rename"));
164
165                         if (f.exists()) {
166                             f.delete();
167                             System.out.println("Supprime : "
168                                     + file.getString("file_name"));
169                         } else {
170                             System.out.println("Existe pas : "
171                                     + file.getString("file_name"));
172                         }
173                     }
174
175                     // Suppression de l'archive du dossier
176                     File archive = new File(Global.UPLOAD_DIR
177                             + Utilities.getFolderArchiveName(
178                                     folder.getString("folder_name"),
179                                     folder.getString("folder_creation_date")));
180
181                     // Envoi du mail à l'auteur du dossier
182                     Mailer.sendMailAuthorExpirationFolder(
183                             folder.getString("user_mail"), modelFolder,
184                             folder.getString("user_langue"));
185
186                     // Envoi du mail au destinataires qui ont accès au dossier
187                     ResultSet recipient = Sql
188                             .query("SELECT * FROM trans_recipients "
189                                     + "LEFT JOIN trans_contacts ON
    PKNoContact=FKNoContact "
190                                     + "WHERE FKNoFolder="
191                                     + folder.getString("PKNoFolder"));
```



```
192
193                     while (recipient.next()) {
194                         Contact contact = new Contact(
195                                 recipient.getInt("PKNoContact"),
196                                 recipient.getString("contact_name"),
197                                 recipient.getString("contact_mail"),
198                                 recipient.getBoolean("contact_internal"));
199
200                         modelFolder.addRecipients(contact);
201                         Mailer.sendMailRecipientExpirationFolder(contact,
202                                 folder.getString("user_mail"), modelFolder);
203                     }
204
205                 }
206
207                 System.out.println(nbDossier + " dossiers affectés");
208
209             } else {
210                 System.out.println("Aucun dossier a archiver");
211             }
212
213         } catch (SQLException e) {
214             e.printStackTrace();
215         }
216     }
217 }
218
```

```java
package toolbox;

import java.io.UnsupportedEncodingException;

/**
 * Class qui répertorie différents outils relatifs à l'utilisation des adresses
 * e-mail<br>
 * <b>Outils principaux :</b><br>
 * <ul>
 * <li>Envoi de mails en java via le protocole SMPT (requis : librairie
 * javamail).</li>
 * <li>Validation et contrôles de formats d'adresses</li>
 * </ul>
 *
 * @author Dominique Roduit
 *
 */
public class Mailer {
    /**
     * Nom ou adresse ip du serveur SMTP
     */
    private final static String SERVER = "localhost";
    /**
     * Suffixe d'une addresse e-mail standard de la CCCVs
     */
    private final static String SUFFIXE_FORMAT = Global
            .getParm("EMAIL_INTERNE_SUFFIXE");
    /**
     * Adresse mail utilisée pour envoyer des messages automatiques aux
     * utilisateurs
     */
    public final static String APP_MAIL_FROM = Global.getParm("APP_MAIL_FROM");
    /**
     * Nom d'affichage de l'adresse e-mail définie par APP_MAIL_FROM
     */
    public final static String APP_MAIL_FROM_NAME = Global
            .getParm("APP_MAIL_FROM_NAME");

    /**
     * Envoi d'un mail au(x) destinataire(s) spécifié(s).
     *
     * @param from
     *            (String) Auteur du mail
     * @param from_name
     *            (String) Nom de l'auteur du mail
     * @param to
     *            (String) Destinataires (séparés par des virgules)
     * @param subject
     *            (String) Sujet du message
     * @param content
     *            (String) Contenu du mail au format HTML
     */
    public static void send(String from, String from_name, String to,
            String subject, String content) {
        try {
            Properties prop = System.getProperties();
            prop.put("mail.smtp.host", SERVER);

            Session session = Session.getInstance(prop);
            Message message = new MimeMessage(session);

            // Auteur
            message.setFrom(new InternetAddress(from, from_name));

            // Destinataire(s)
            InternetAddress[] internetAddresses;
            if (to.indexOf(",") > -1) {
                String[] to_array = to.split(",");
                internetAddresses = new InternetAddress[to_array.length + 1];
                for (int i = 0; i < to_array.length; i++) {
                    internetAddresses[i] = new InternetAddress(to_array[i]);
                }
            } else {
                internetAddresses = new InternetAddress[1];
                internetAddresses[0] = new InternetAddress(to);
            }
            message.setRecipients(Message.RecipientType.TO, internetAddresses);

            // Sujet
            message.setSubject(new String(subject.getBytes(), "UTF-8"));

            // Contenu du mail
            message.setContent(new String(content.getBytes(), "UTF-8"),
                    "text/html; charset=UTF-8");

            message.setSentDate(new Date());
            session.setDebug(true);

            Transport.send(message);
        } catch (NoSuchProviderException e) {
            System.err.println("Pas de transport disponible pour ce protocole");
            System.err.println(e);
        } catch (AddressException e) {
            System.err.println("Adresse invalide");
            System.err.println(e);
        } catch (MessagingException e) {
            System.err.println("Erreur dans le message");
            System.err.println(e);
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }

    /**
     * Envoi d'un mail automatique, l'adresse de l'auteur reste fixe et le nom
     * aussi
     *
     * @param to
     *            Destinataires séparés par des virgules
     * @param subject
     *            Sujet du message
     * @param content
     *            Contenu du mail
     */
    public static void send(String to, String subject, String content) {
        send(APP_MAIL_FROM, APP_MAIL_FROM_NAME, to, subject, content);
    }

    /**
     * Mail envoyé à l'auteur d'un dossier avant la suppression de celui-ci.
     *
     * @param to
     *            Adresse e-mail de l'auteur du dossier
     * @param folder
```

```
144     *            Informations sur le dossier
145     * @param langue
146     *            Langue de l'utilisateur
147     */
148    public static void sendMailAuthorBeforeRemovingFolder(String to,
149            Folder folder, String langue) {
150        String history = getHistoryForContactsOfFolder(folder, langue);
151
152        // Contenu du mail
153        String mailContent = Global.getParm("MAIL_HEADER_CONTENT")
154                + Global.getParm("MAIL_AUTHOR_BEFORE_REMOVING_CONTENT")
155                        .replace("%nom%", folder.getName())
156                        .replace(
157                                "%date%",
158                                Utilities.formatSQLDate(
159                                        folder.getExpiration_date(), "dd.MM.YY"))
160                        .replace(
161                                "%heure%",
162                                Utilities.zeroFill(Integer.parseInt(Utilities
163                                        .formatSQLDate(
164                                                folder.getExpiration_date(),
165                                                "HH")) + 1)
166                                        + ":00")
167                        .replace("%historique%", history)
168                + Global.getParm("MAIL_FOOTER_CONTENT");
169
170        Mailer.send(to,
171                Global.i("SUBJECT_MAIL_AUTHOR_BEFORE_REMOVING_FOLDER", langue),
172                mailContent);
173    }
174
175    /**
176     * Mail envoyé à chacun des contacts qui avaient accès au dossier expiré
177     *
178     * @param to
179     *            Adresse e-mail du contact
180     * @param folder
181     *            Informations sur le dossier
182     * @param langue
183     *            Langue de l'utilisateur
184     */
185    public static void sendMailRecipientBeforeRemovingFolder(Contact contact,
186            String mail_author, Folder folder) {
187        String langue = "fr";
188
189        // Actions journalisées pour ce contact
190        String history = getHistoryOfContactFromFolder(contact, folder, langue);
191
192        // Contenu du mail
193        String mailContent = Global.getParm("MAIL_HEADER_CONTENT")
194                + Global.getParm("MAIL_RECIPIENT_BEFORE_REMOVING_CONTENT")
195                        .replace(
196                                "%date%",
197                                Utilities.formatSQLDate(
198                                        folder.getExpiration_date(), "dd.MM.YY"))
199                        .replace("%author%", mail_author)
200                        .replace("%nom%", folder.getName())
201                        .replace(
202                                "%heure%",
203                                Utilities.zeroFill(Integer.parseInt(Utilities
204                                        .formatSQLDate(
205                                                folder.getExpiration_date(),
```



```
206                                                "HH")) + 1)
207                                        + ":00")
208                        .replace("%historique%", history)
209                + Global.getParm("MAIL_FOOTER_CONTENT");
210
211        Mailer.send(contact.getMail(), Global.i(
212                "SUBJECT_MAIL_RECIPIENT_BEFORE_REMOVING_FOLDER", langue),
213                mailContent);
214    }
215
216    /**
217     * Mail envoyé à l'auteur d'un dossier lors de la suppression des fichiers
218     * et l'expiration de celui-ci.
219     *
220     * @param to
221     *            Adresse e-mail de l'auteur du dossier
222     * @param modelFolder
223     *            Informations sur le dossier
224     * @param langue
225     *            Langue de l'utilisateur
226     */
227    public static void sendMailAuthorExpirationFolder(String to, Folder folder,
228            String langue) {
229        // Actions journalisées pour ce contact
230        String history = getHistoryForContactsOfFolder(folder, langue);
231
232        // Contenu du mail
233        String mailContent = Global.getParm("MAIL_HEADER_CONTENT")
234                + Global.getParm("MAIL_AUTHOR_EXPIRATION_FOLDER_CONTENT")
235                        .replace("%nom%", folder.getName())
236                        .replace(
237                                "%date%",
238                                Utilities.formatSQLDate(
239                                        folder.getExpiration_date(), "dd.MM.YY"))
240                        .replace("%heure%", Utilities.getDate("HH") + ":00")
241                        .replace("%historique%", history)
242                + Global.getParm("MAIL_FOOTER_CONTENT");
243
244        Mailer.send(to,
245                Global.i("SUBJECT_MAIL_AUTHOR_EXPIRATION_FOLDER", langue),
246                mailContent);
247    }
248
249    /**
250     * Mail envoyé à chacun des contacts qui avaient accès au dossier expiré
251     *
252     * @param to
253     *            Adresse e-mail du contact
254     * @param folder
255     *            Informations sur le dossier
256     * @param langue
257     *            Langue de l'utilisateur
258     */
259    public static void sendMailRecipientExpirationFolder(Contact contact,
260            String mail_author, Folder folder) {
261        String langue = "fr";
262
263        // Actions journalisées pour ce contact
264        String history = getHistoryOfContactFromFolder(contact, folder, langue);
265
266        // Contenu du mail
267        String mailContent = Global.getParm("MAIL_HEADER_CONTENT")
```

```
268                 + Global.getParm("MAIL_RECIPIENT_EXPIRATION_FOLDER_CONTENT")
269                         .replace("%nom%", contact.getName())
270                         .replace(
271                                 "%date%",
272                                 Utilities.formatSQLDate(
273                                         folder.getExpiration_date(), "dd.MM.YY"))
274                         .replace("%author%", mail_author)
275                         .replace("%dossier%", folder.getName())
276                         .replace("%heure%", Utilities.getDate("HH") + ":00")
277                         .replace("%historique%", history)
278                 + Global.getParm("MAIL_FOOTER_CONTENT");
279 
280         Mailer.send(contact.getMail(),
281                 Global.i("SUBJECT_MAIL_RECIPIENT_EXPIRATION_FOLDER", langue),
282                 mailContent);
283     }
284 
285     /**
286      * Retourne l'historiques de tous les contacts d'un dossier. L'historique
287      * Résumé et détaillé.
288      *
289      * @param folder
290      *            Modèle du dossier de référence
291      * @param langue
292      *            Langue de l'utilisateur
293      * @return Historique complet de tous les contacts d'un dossier
294      */
295     private static String getHistoryForContactsOfFolder(Folder folder,
296             String langue) {
297         String history = "";
298 
299         // Sélection des contacts qui ont accès au dossier
300         ResultSet contact = Sql.query("SELECT * FROM trans_folders "
301                 + "LEFT JOIN trans_recipients ON FKNoFolder=PKNoFolder "
302                 + "LEFT JOIN trans_contacts ON FKNoContact=PKNoContact "
303                 + "WHERE PKNoFolder=" + folder.getPKNoFolder());
304 
305         // Historique résumé
306         try {
307             history = "<br><h3>" + Global.i("CAPTION_RESUME_HISTORY", langue)
308                     + "</h3><table border=\"0\">";
309             while (contact.next()) {
310                 ResultSet action = Sql
311                         .query("SELECT *, COUNT(*) AS NB FROM trans_journal_folders
    "
312                                 + "LEFT JOIN trans_files ON FKNoFile=PKNoFile "
313                                 + "WHERE FKNoContact="
314                                 + contact.getString("PKNoContact")
315                                 + " AND trans_journal_folders.FKNoFolder="
316                                 + folder.getPKNoFolder()
317                                 + " "
318                                 + "GROUP BY FKNoFile, joufo_action "
319                                 + "ORDER BY FKNoFile, joufo_date DESC");
320 
321                 history += "<tr><td colspan=\"5\" style=\"background:#000;
    color:#fff; font-weight: bold;\">"
322                         + contact.getString("contact_name") + "</td></tr>";
323 
324                 // S'il existe des actions pour ce contact
325                 if (action.first()) {
326                     int i = 0;
327                     action.beforeFirst();
```



```
328 
329                     while (action.next()) {
330                         i++;
331 
332                         String color = "ddf";
333                         String name = action.getString("file_rename");
334                         if (action.getString("FKNoFile") == null) {
335                             if (action.getString("joufo_action").equals(
336                                     "download")) {
337                                 name = Global.i("CAPTION_ARCHIVE_OF_FOLDER",
338                                         langue);
339                                 color = "fdd";
340                             } else {
341                                 name = Global.i("CAPTION_CONSULTATION", langue);
342                                 color = "dfd";
343                             }
344                         }
345 
346                         history += "<tr style=\"background:#" + color + "\">"
347                                 + "<td>" + i + " </td>" + "<td>   </td>"
348                                 + "<td>" + name + "</td>" + "<td>   </td>"
349                                 + "<td>" + action.getString("NB") + "x</td>"
350                                 + "</tr>";
351                     }
352                 } else {
353                     history += "<tr>" + "<td colspan=\"5\">"
354                             + Global.i("CAPTION_NO_ACCES_AND_DOWNLOAD", langue)
355                             + "</td>" + "</tr>";
356                 }
357 
358             }
359             history += "</table>";
360         } catch (SQLException e) {
361             e.printStackTrace();
362         }
363 
364         // Historique détaillé
365         try {
366             history +=
    "<br>_____<br><br>"
367                     + "<h3>"
368                     + Global.i("CAPTION_EXPAND_HISTORY", langue)
369                     + "</h3><table border=\"0\">";
370             contact.beforeFirst();
371             while (contact.next()) {
372                 // Sélection des actions pour chaques contacts
373                 ResultSet action = Sql
374                         .query("SELECT * FROM trans_journal_folders "
375                                 + "LEFT JOIN trans_files ON FKNoFile=PKNoFile "
376                                 + "WHERE FKNoContact="
377                                 + contact.getString("PKNoContact")
378                                 + " AND trans_journal_folders.FKNoFolder="
379                                 + folder.getPKNoFolder() + " "
380                                 + "ORDER BY joufo_date DESC");
381 
382                 history += "<tr><td colspan=\"7\" style=\"background:#000;
    color:#fff; font-weight: bold;\">"
383                         + contact.getString("contact_name") + "</td></tr>";
384 
385                 // S'il existe des actions pour ce contact
386                 if (action.first()) {
387                     int i = 0;
```

```
388                     action.beforeFirst();
389
390                     while (action.next()) {
391                         i++;
392                         String actionName = "";
393                         String actionDescription = "";
394                         String actionDate = Utilities.formatSQLDate(
395                                 action.getString("joufo_date"),
396                                 "dd.MM.YY HH:mm");
397
398                         if (action.getString("joufo_action").equals("download")) {
399                             actionName = Global
400                                     .i("CAPTION_DOWNLOADING", langue);
401                             actionDescription = (action.getString("FKNoFile") !=
    null) ? "<span style=\"color:blue\">"
402                                     + action.getString("file_rename")
403                                     + "</span>"
404                                     : "<span style=\"color:red\">"
405                                             + Global.i(
406                                                     "CAPTION_ARCHIVE_OF_FOLDER",
407                                                     langue) + "</span>";
408                         } else {
409                             actionName = Global.i("CAPTION_CONSULTATION",
410                                     langue);
411                             actionDescription = "<span style=\"color:green\">"
412                                     + Global.i("CAPTION_FOLDER_CONSULTATION",
413                                             langue) + "</span>";
414                         }
415
416                         history += "<tr>" + "<td>" + i + " </td>"
417                                 + "<td>   </td>" + "<td>" + actionName
418                                 + "</td>" + "<td>   </td>" + "<td>"
419                                 + actionDescription + "</td>"
420                                 + "<td>   </td>" + "<td>" + actionDate
421                                 + "</td>" + "</tr>";
422
423                     }
424                 } else {
425                     history += "<tr><td colspan=\"5\">"
426                             + Global.i("CAPTION_NO_ACCES_AND_DOWNLOAD", langue)
427                             + "</td></tr>";
428                 }
429             }
430             history += "</table>";
431         } catch (SQLException e) {
432             e.printStackTrace();
433         }
434
435         if (history.equals("")) {
436             history = "- " + Global.i("CAPTION_EMPTY_HISTORY", langue) + " -";
437         }
438         return history;
439     }
440
441     /**
442      * Retourne l'historique d'un contact sur un dossier
443      *
444      * @param contact
445      *            Modèle du contact de référence
446      * @param folder
447      *            Modèle du dossier de référence
448      * @param langue
```



```
449      *            Langue de l'utilisateur
450      * @return Historique du contact mentionné
451      */
452     private static String getHistoryOfContactFromFolder(Contact contact,
453             Folder folder, String langue) {
454         String history = "";
455         ResultSet action = Sql.query("SELECT * FROM trans_journal_folders "
456                 + "LEFT JOIN trans_files ON FKNoFile=PKNoFile "
457                 + "WHERE FKNoContact=" + contact.getPK()
458                 + " AND trans_journal_folders.FKNoFolder="
459                 + folder.getPKNoFolder() + " " + "ORDER BY joufo_date DESC");
460
461         try {
462             if (action.first()) {
463                 int consultation = 0;
464                 int i = 0;
465                 action.beforeFirst();
466                 history = "<br><table border=\"0\"><tr><th> <th></th> </th><th><b>"
467                         + Global.i("CAPTION_FILE_DOWNLOADED", langue)
468                         + "</b></th> <th></th> <th><b>"
469                         + Global.i("CAPTION_DATE", langue) + "</b></th></tr>";
470                 while (action.next()) {
471                     String actionDescription = "";
472                     String actionDate = Utilities.formatSQLDate(
473                             action.getString("joufo_date"), "dd.MM.YY HH:mm");
474
475                     if (action.getString("joufo_action").equals("download")) {
476                         i++;
477                         actionDescription = (action.getString("FKNoFile") != null)
    ? "<span style=\"color:blue\">"
478                                 + action.getString("file_rename") + "</span>"
479                                 : "<span style=\"color:red\">"
480                                         + Global.i("CAPTION_ARCHIVE_OF_FOLDER",
481                                                 langue) + "</span>";
482
483                         history += "<tr>" + "<td>" + i + " </td>"
484                                 + "<td>   </td>" + "<td>"
485                                 + actionDescription + "</td>"
486                                 + "<td>   </td>" + "<td>" + actionDate
487                                 + "</td>" + "</tr>";
488                     } else {
489                         consultation++;
490                     }
491
492                 }
493                 history += "</table>";
494
495                 String resume = "<strong>"
496                         + Global.i("CAPTION_CONSULTATION", langue) + " : "
497                         + consultation + "x</strong>";
498
499                 history = resume + "<br>" + history;
500             }
501         } catch (SQLException e) {
502             e.printStackTrace();
503         }
504
505         if (history.equals("")) {
506             history = "- " + Global.i("CAPTION_EMPTY_HISTORY", langue) + " -";
507         }
508         return history;
509     }
```